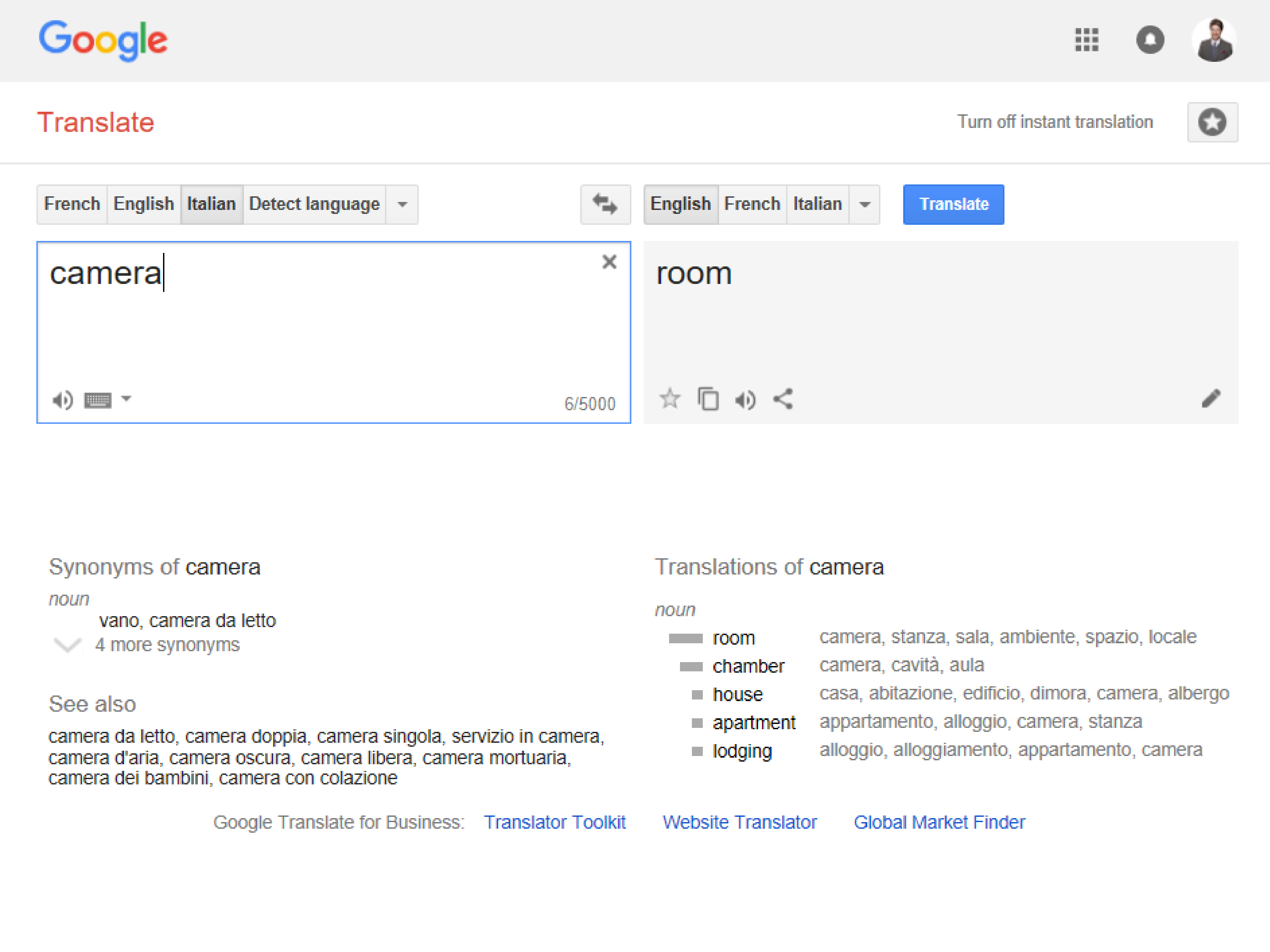
ECE 4973: Lecture 4 Camera models and calibration

Samuel Cheng

Slide credit: James Tompkin, Naoh Snavely

What is a camera?



camera

room

Synonyms of camera

noun

vano, camera da letto

4 more synonyms

See also

camera da letto, camera doppia, camera singola, servizio in camera, camera d'aria, camera oscura, camera libera, camera mortuaria, camera dei bambini, camera con colazione

Translations of camera

noun

room	camera, stanza, sala, ambiente, spazio, locale
chamber	camera, cavità, aula
house	casa, abitazione, edificio, dimora, camera, albergo
apartment	appartamento, alloggio, camera, stanza
lodging	alloggio, alloggiamento, appartamento, camera

Camera obscura: dark room

- Known during classical period in China and Greece (e.g., Mo-Ti, China, 470BC to 390BC)

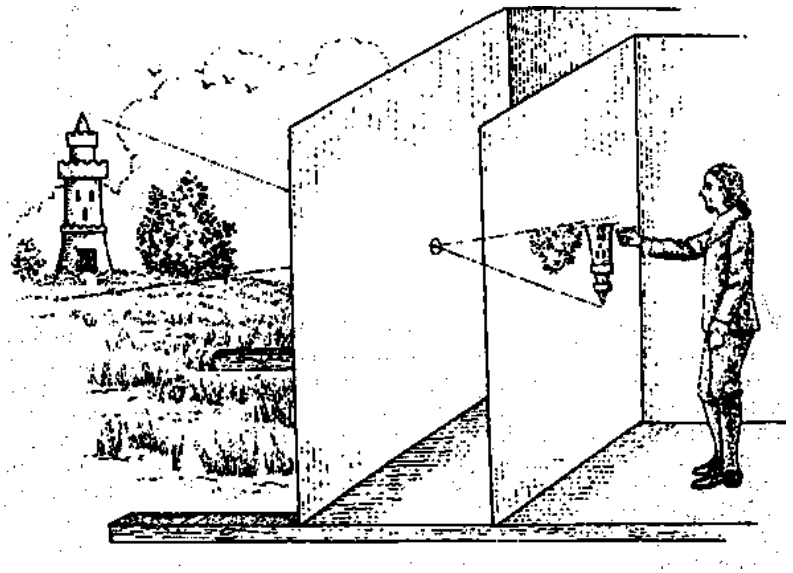


Illustration of Camera Obscura

Freestanding camera obscura at UNC Chapel Hill

Photo by Seth Ilys

James Hays

Camera obscura / lucida used for tracing

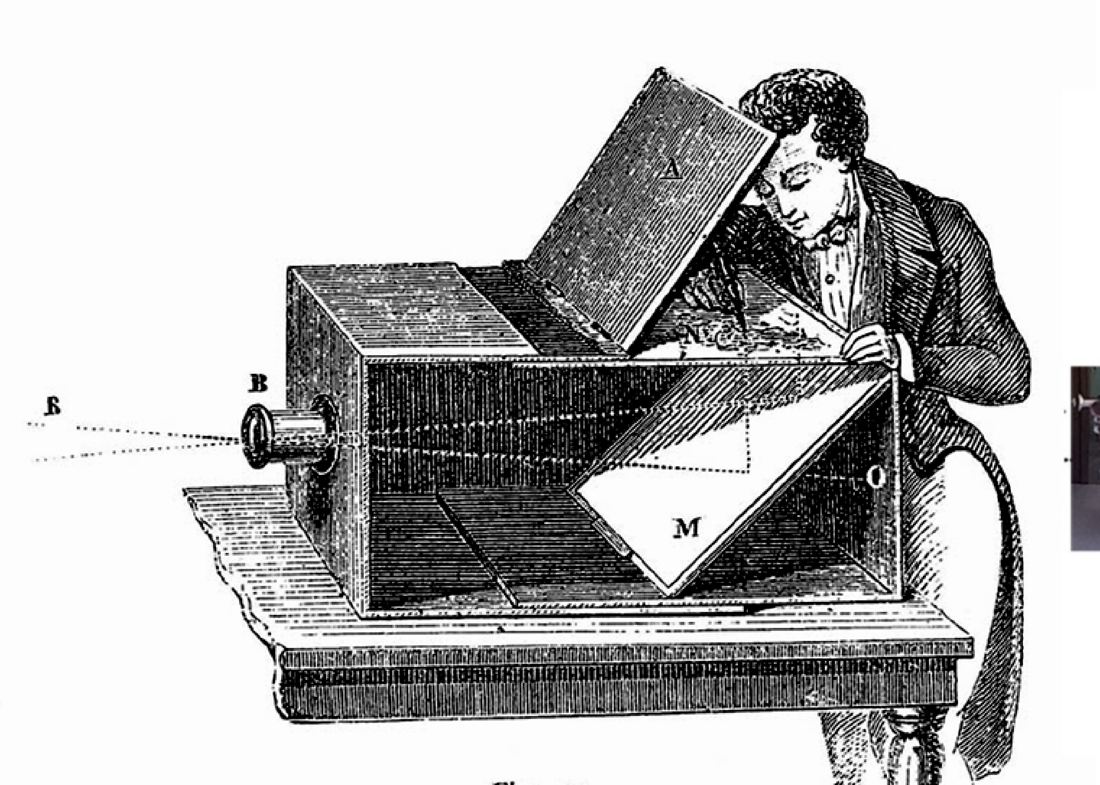
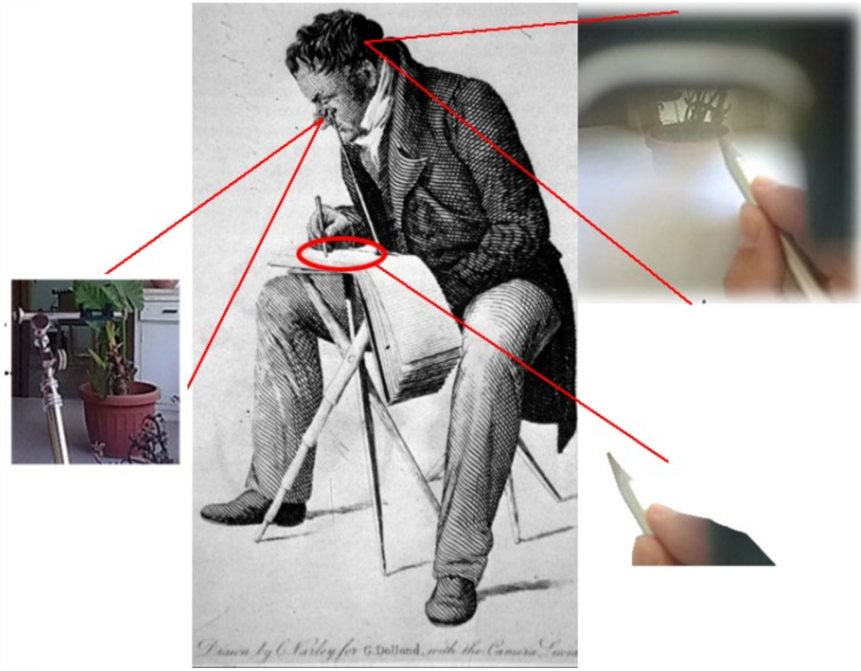


Fig. 434

Lens Based Camera Obscura, 1568



Camera lucida

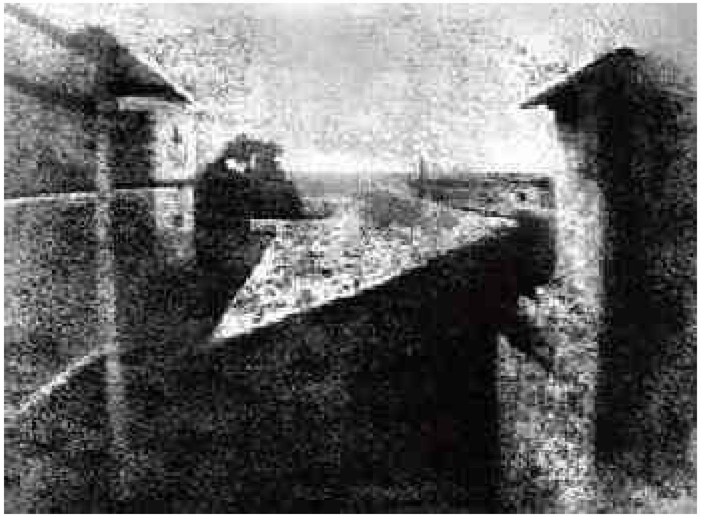
drawingchamber.wordpress.com

First Photograph

Oldest surviving photograph

- Took 8 hours on pewter plate

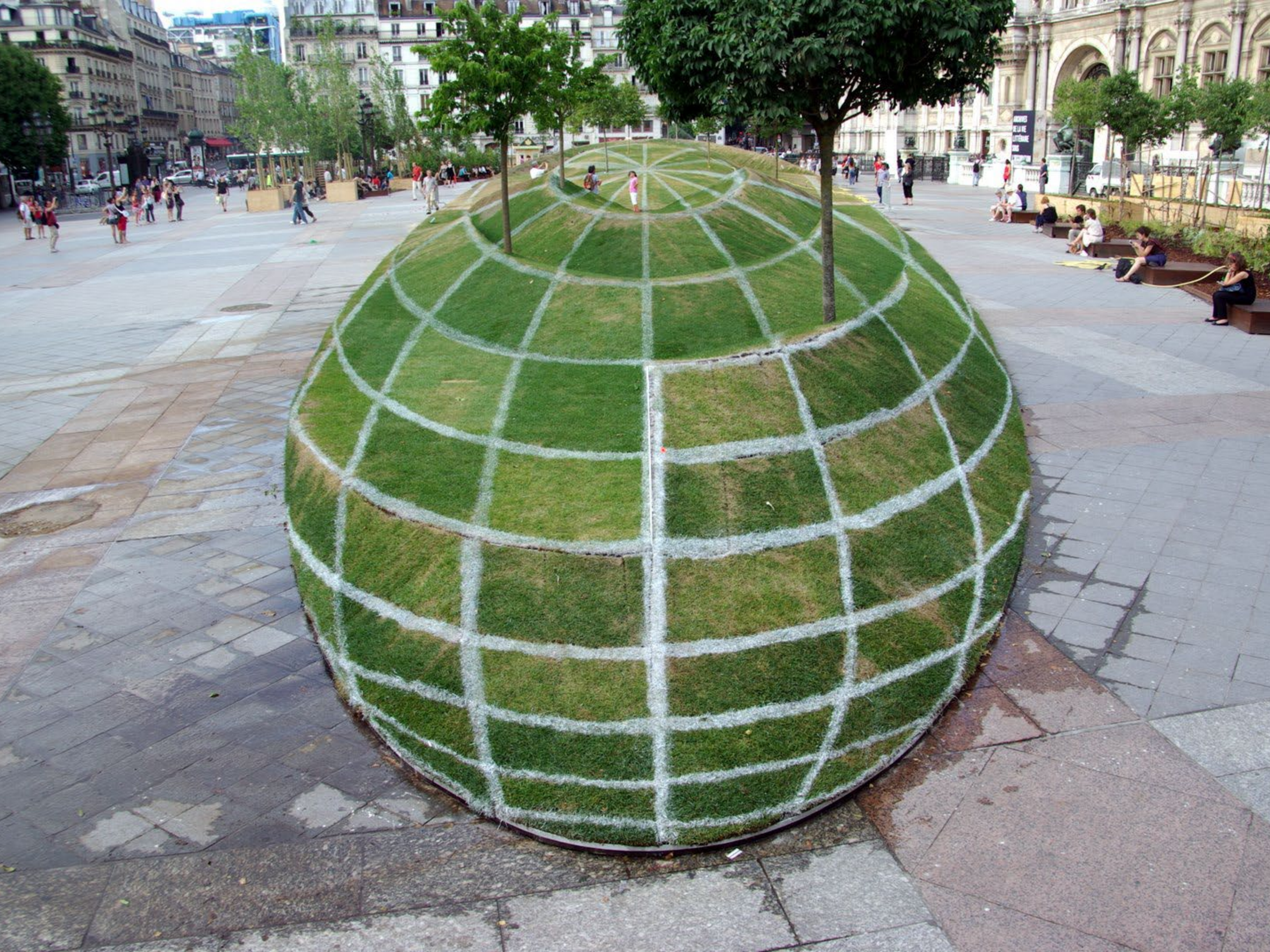
Photograph of the first photograph



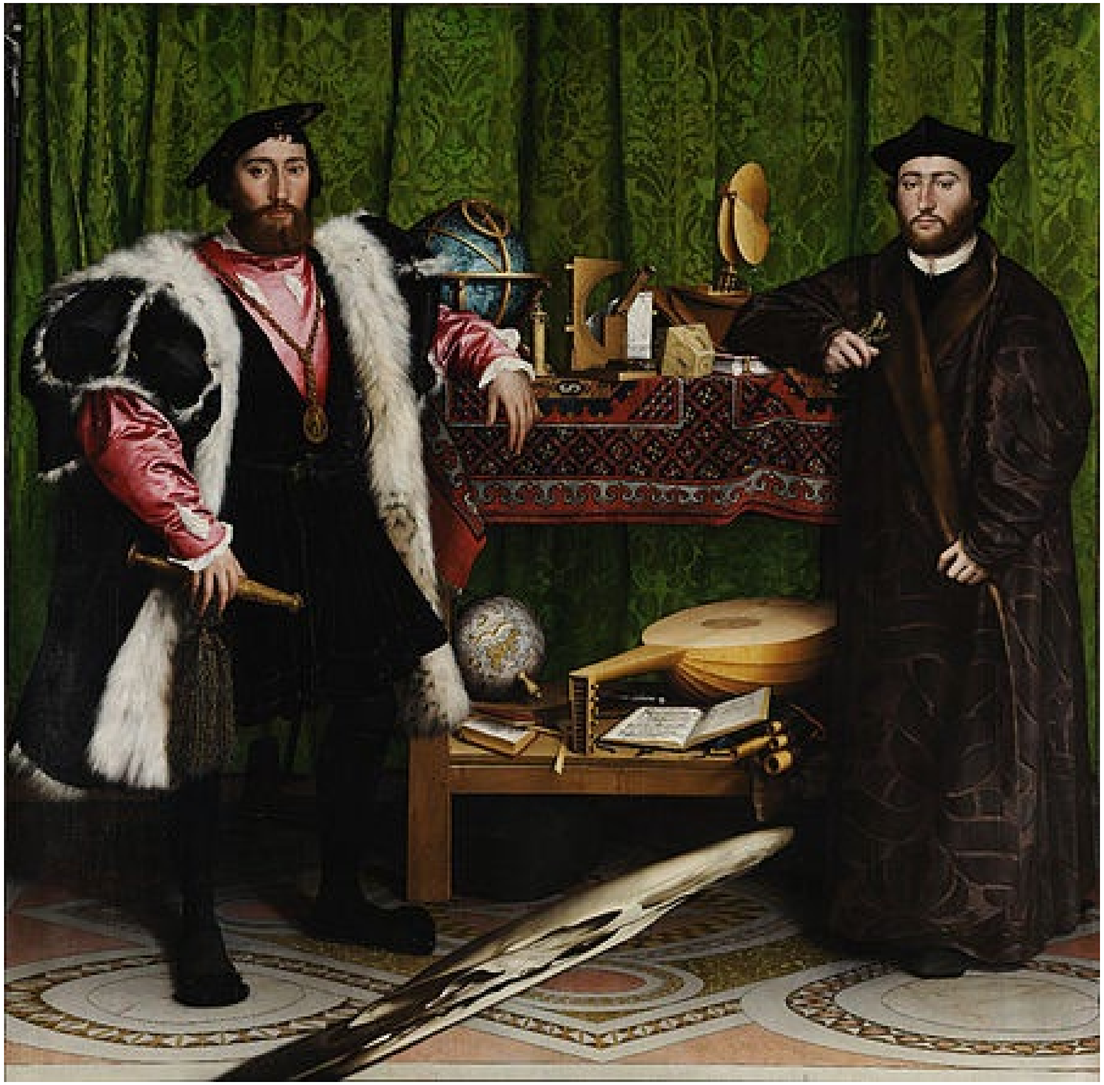
Joseph Niepce, 1826

Stored at UT Austin

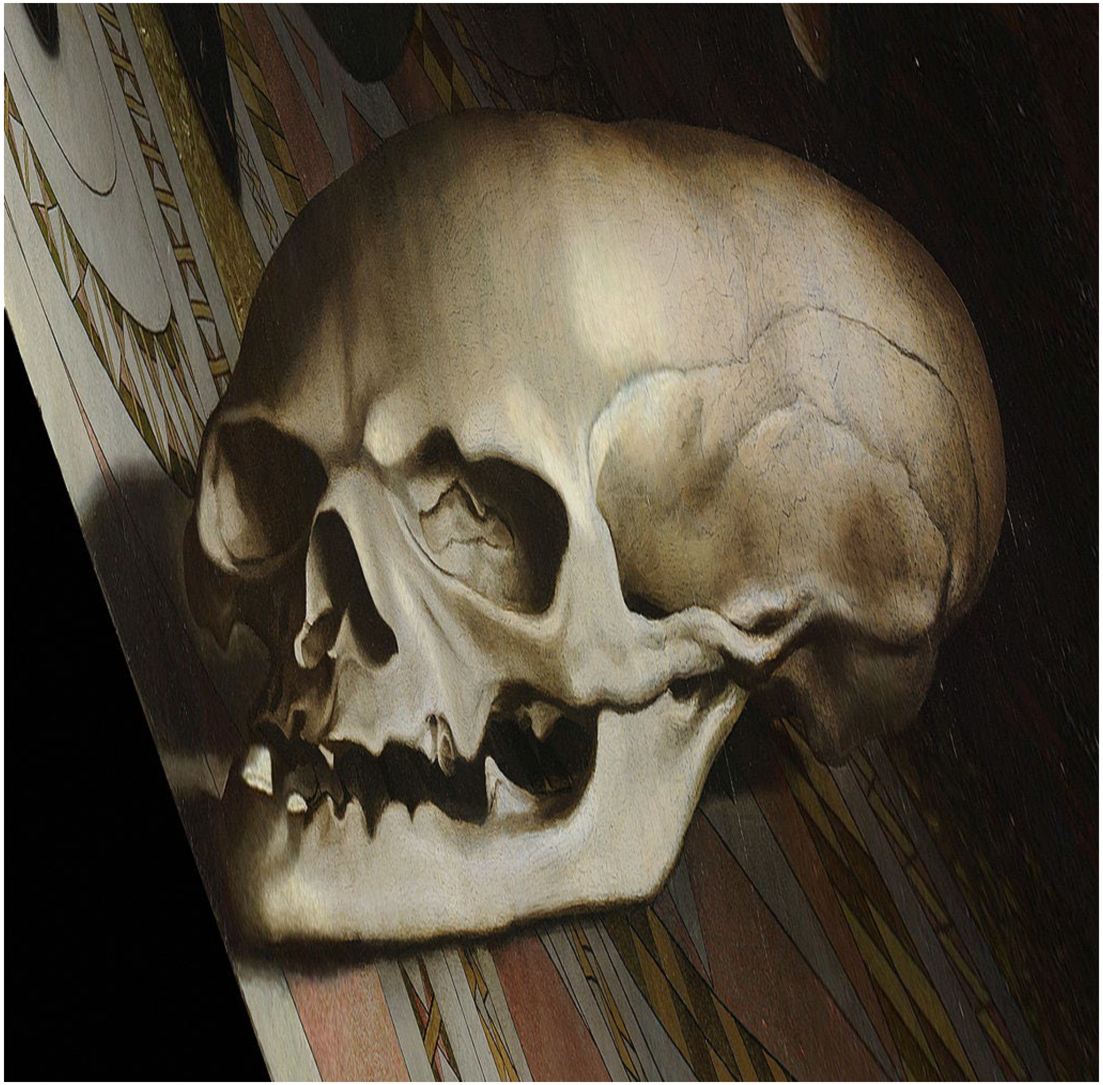
Niepce later teamed up with Daguerre, who eventually created Daguerrotypes



Holbein’s The Ambassadors - 1533



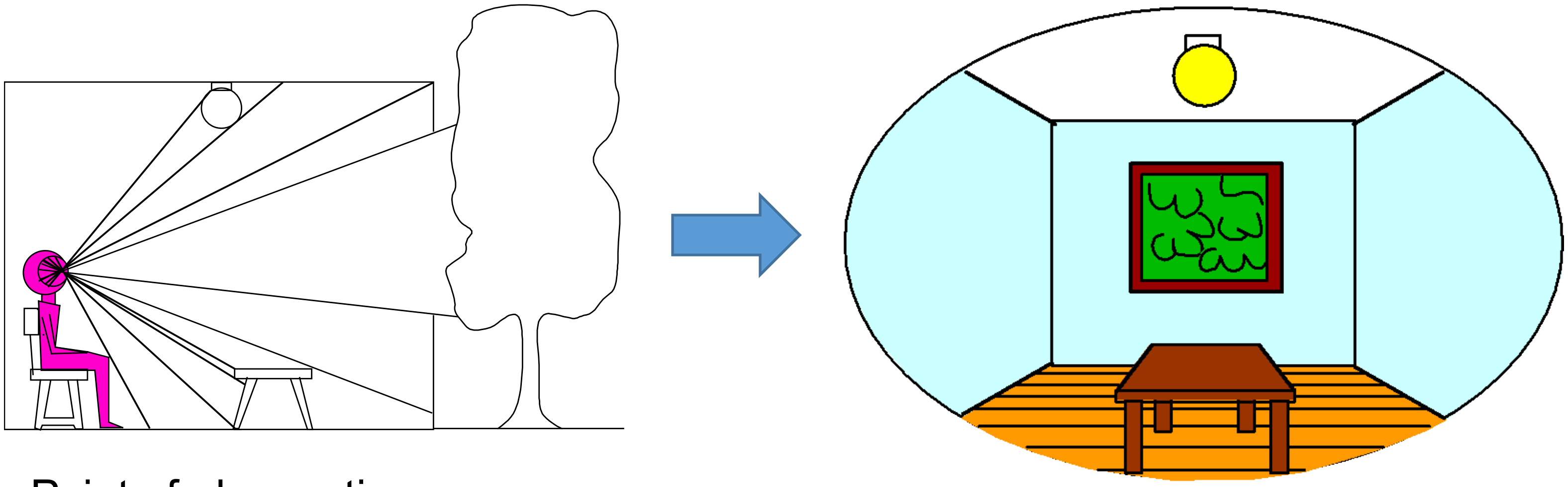
Holbein’s The Ambassadors – Memento Mori



Dimensionality Reduction Machine (3D to 2D)

3D world

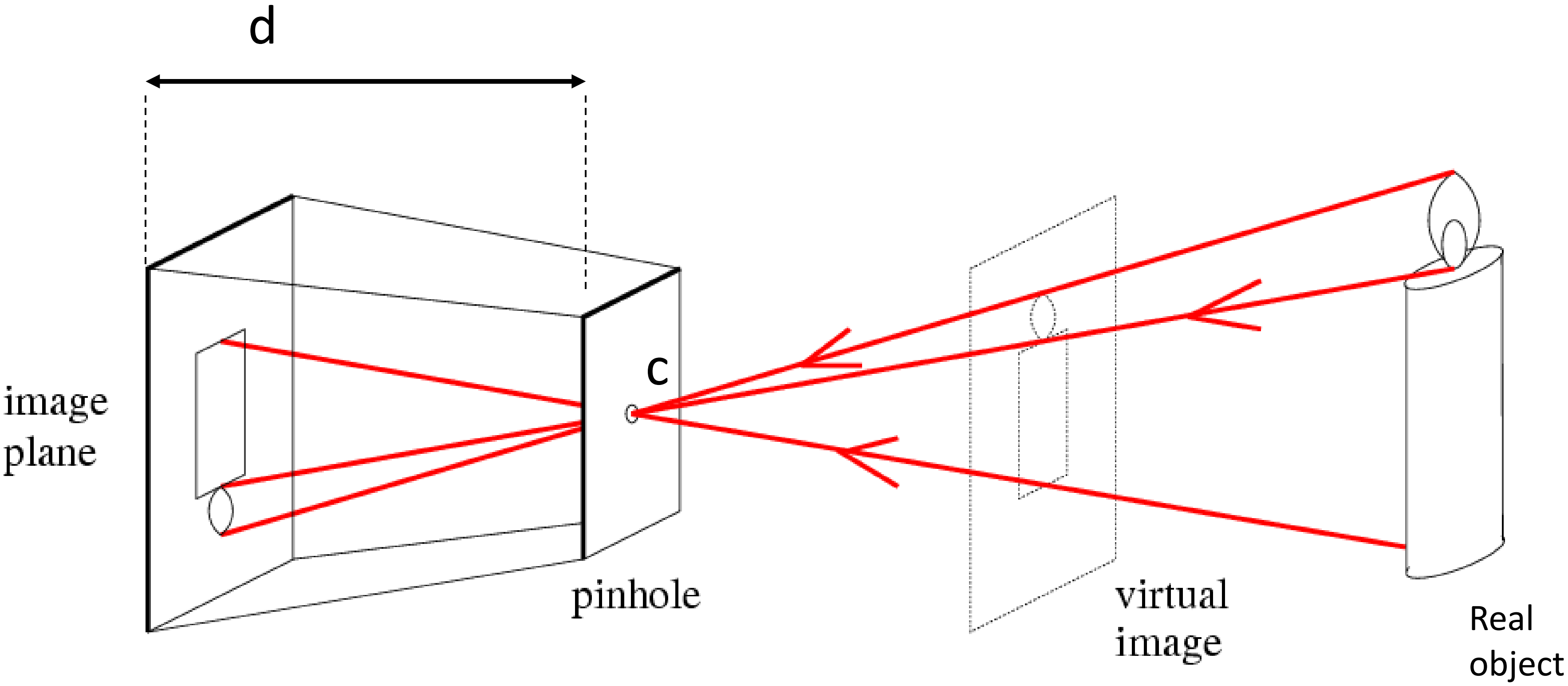
2D image



Point of observation

Figures © Stephen E. Palmer, 2002

Pinhole camera model

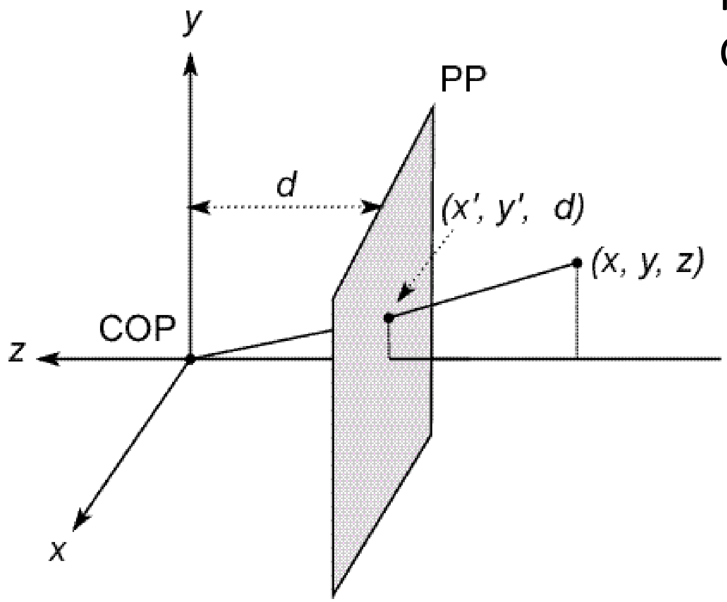


d = “Focal length” (or f)

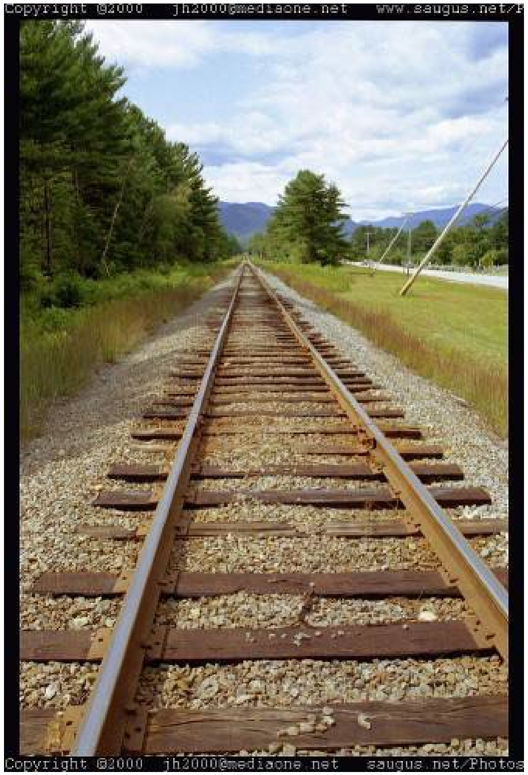
c = Optical center of the camera

Figure from Forsyth

Modeling projection



PP: projection plane
COP: center of projection


Copyright ©2000 jh2000@mediaone.net www.saugus.net/P
Copyright ©2000 jh2000@mediaone.net saugus.net/Photos

- Both (x',y',d) and (x,y,z) project to the same point at
- (x,y,z) -> (x',y') where x' = d (x/z) and y' = d (y/z)
- Magnification = d/z

Modeling projection

- Is the projection a linear transformation?
  - no—division by z is nonlinear

Homogeneous coordinates to the rescue!

Recall that:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/d \end{bmatrix} \Rightarrow \left(d\frac{x}{z}, d\frac{y}{z} \right)$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**

Perspective Projection

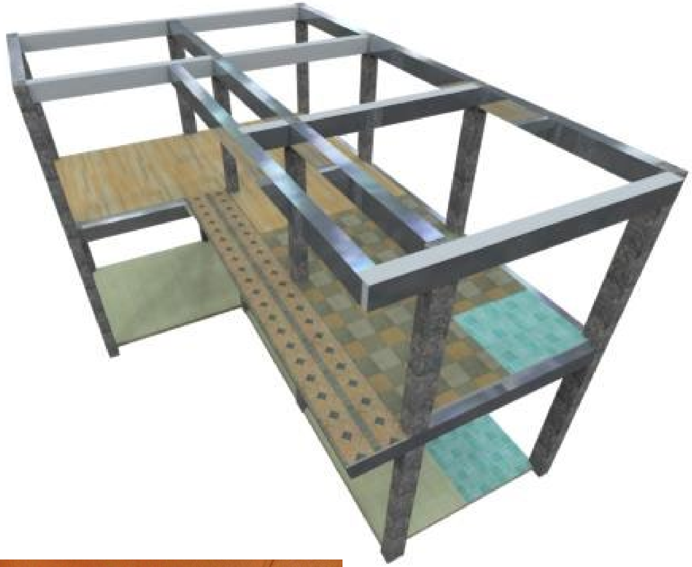
- Note that scaling the projection matrix does not change the transform

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/d \end{bmatrix} \Rightarrow \left(d\frac{x}{z}, d\frac{y}{z} \right)$$

$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ z \end{bmatrix} \Rightarrow \left(d\frac{x}{z}, d\frac{y}{z} \right)$$

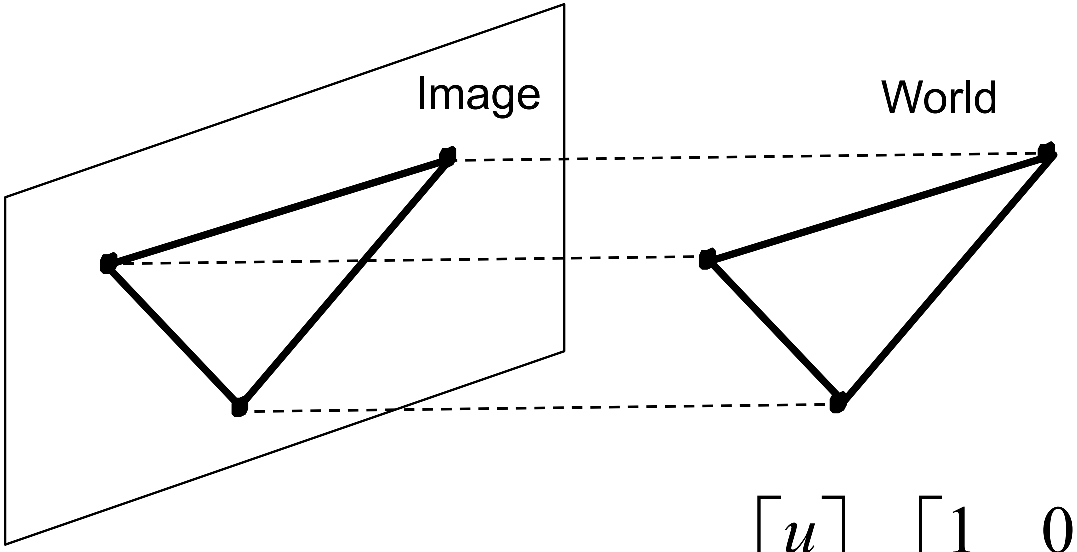
Perspective projection

EVERYONE
E
ESRB
SIMCITY 4



Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the image plane is infinite

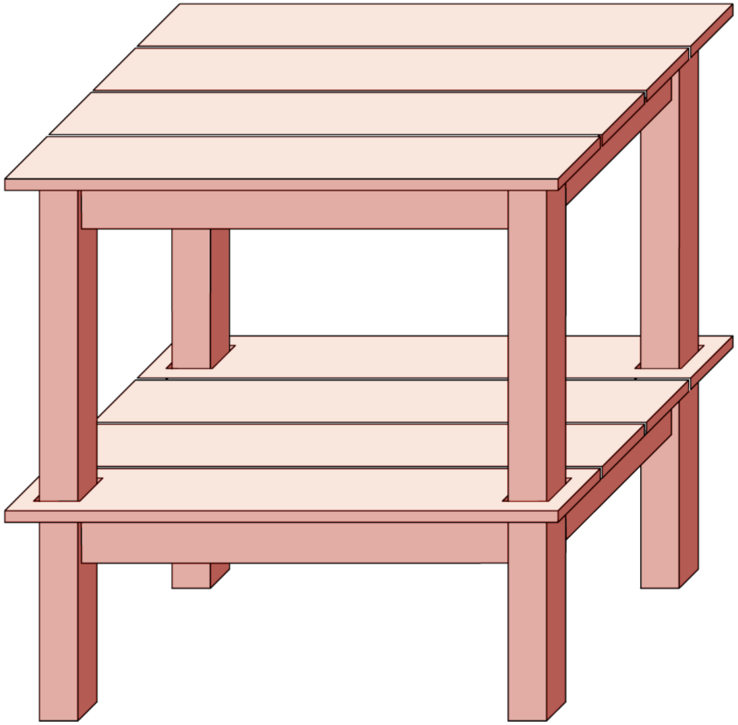


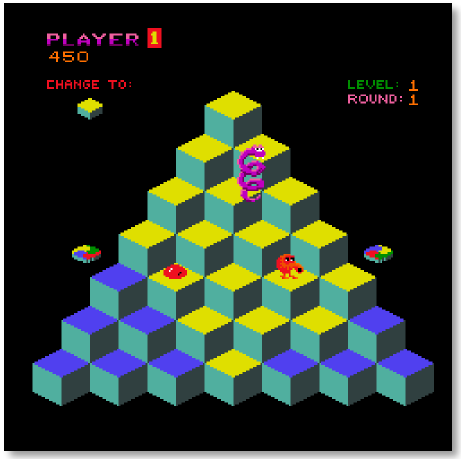
  - Also called "parallel projection"
  - What's the projection matrix?

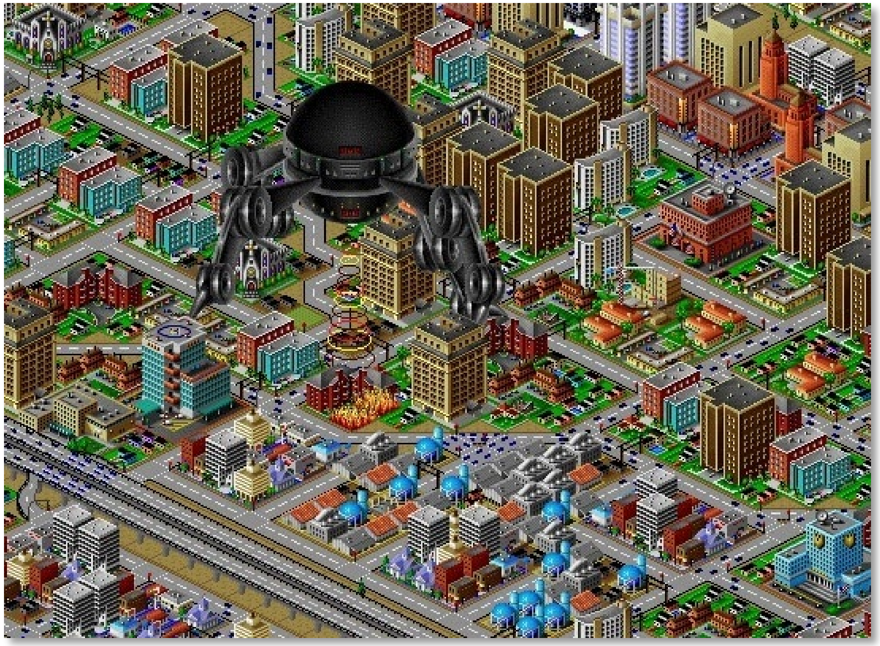
$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slide by Steve Seitz

Orthographic projection



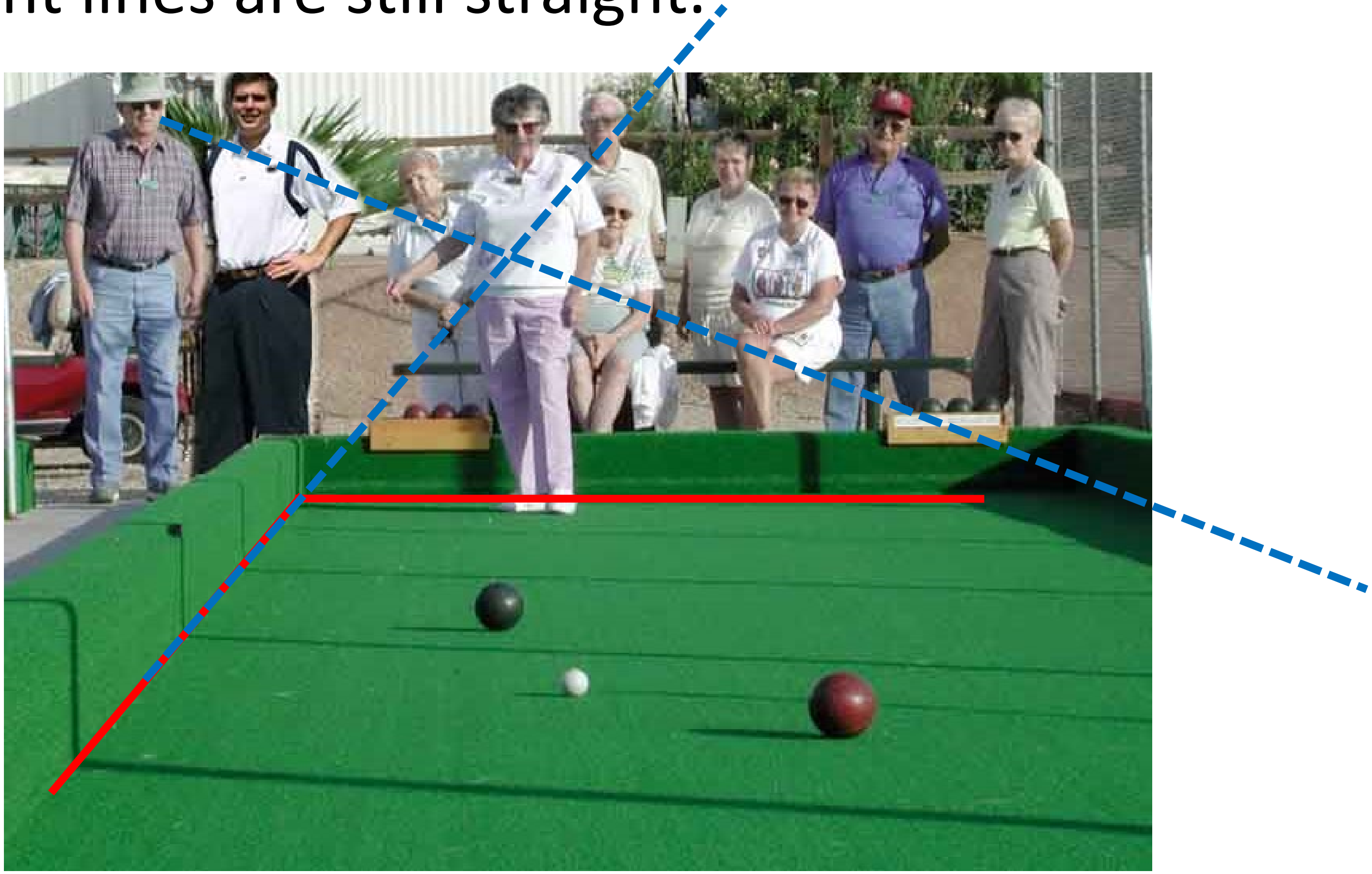




Perspective projection

What is preserved?

- Straight lines are still straight.



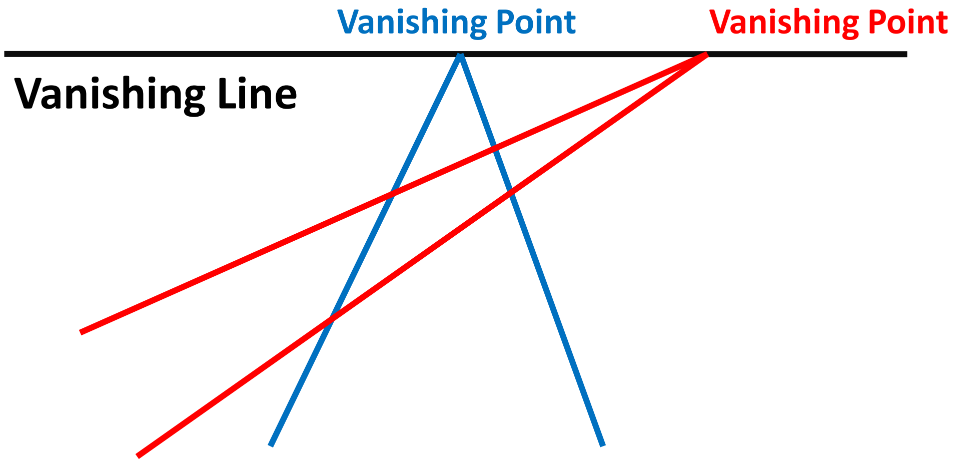
Vanishing points and lines

Parallel lines in the world

intersect in the projected image at a “vanishing point”.

Parallel lines on the same plane in the world converge to vanishing points on a “vanishing line”.

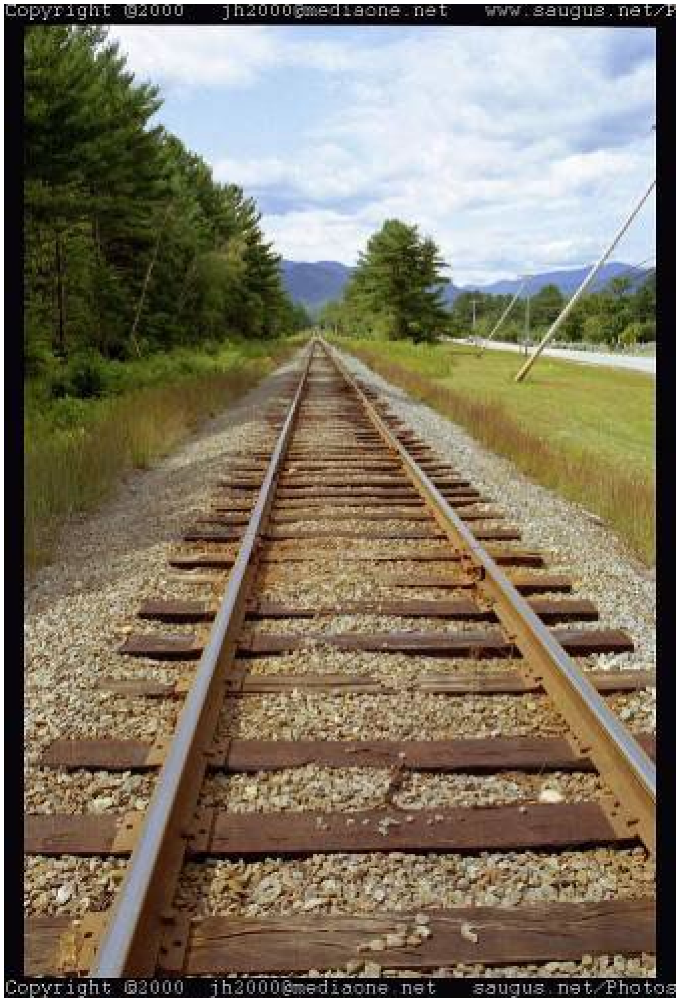
E.G., the horizon.



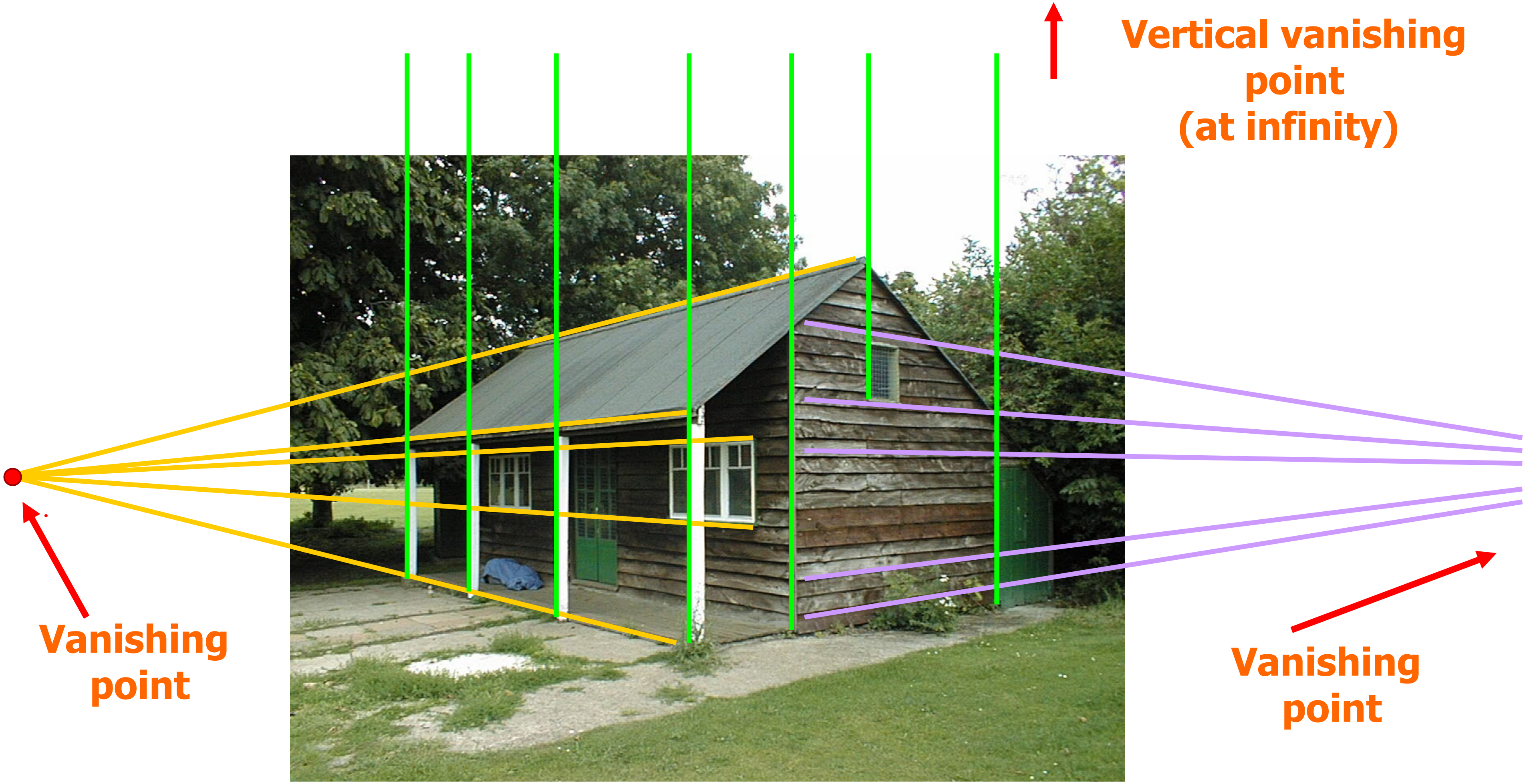
Vanishing Point

Vanishing Point

Vanishing Line


Copyright @2000 jh2000@mediaone.net www.saugus.net/P
Copyright @2000 jh2000@mediaone.net saugus.net/Photos

Vanishing points and lines



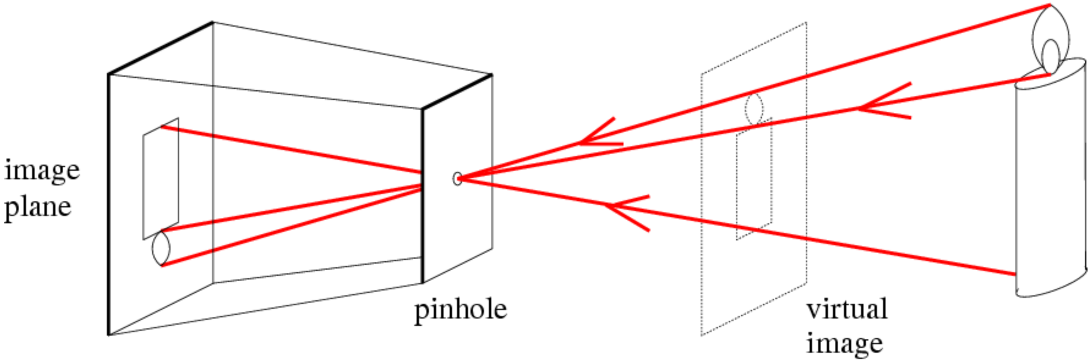
Slide from Efros, Photo from Criminisi

Why parallel lines vanishing to a point

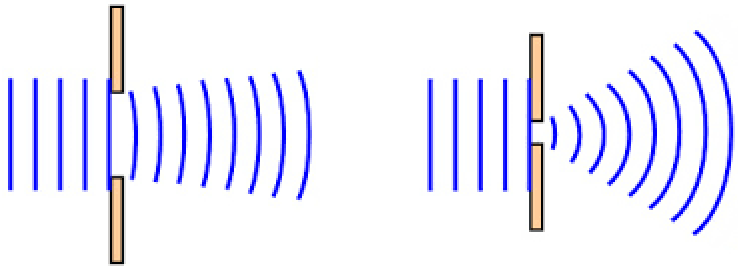
- Consider parallel lines $\begin{pmatrix} x \\ y \\ z \end{pmatrix} + t \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} + \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$ with different shift $\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$
- They project to $\left(\frac{x+t\Delta x+s_x}{z+t\Delta z+s_z} d, \frac{y+t\Delta y+s_y}{z+t\Delta z+s_z} d\right)$ and converge to a single point $\left(\frac{\Delta x}{\Delta z} d, \frac{\Delta y}{\Delta z} d\right)$ as $t \rightarrow \infty$ (except $\Delta z = 0$)

Projection properties

- Many-to-one: any points along same ray map to same point in image
- Points → points
- Lines → lines
- But line through focal point projects to a point
- Planes → planes (or half-planes)
  - But plane through focal point projects to line



Size of the pinhole

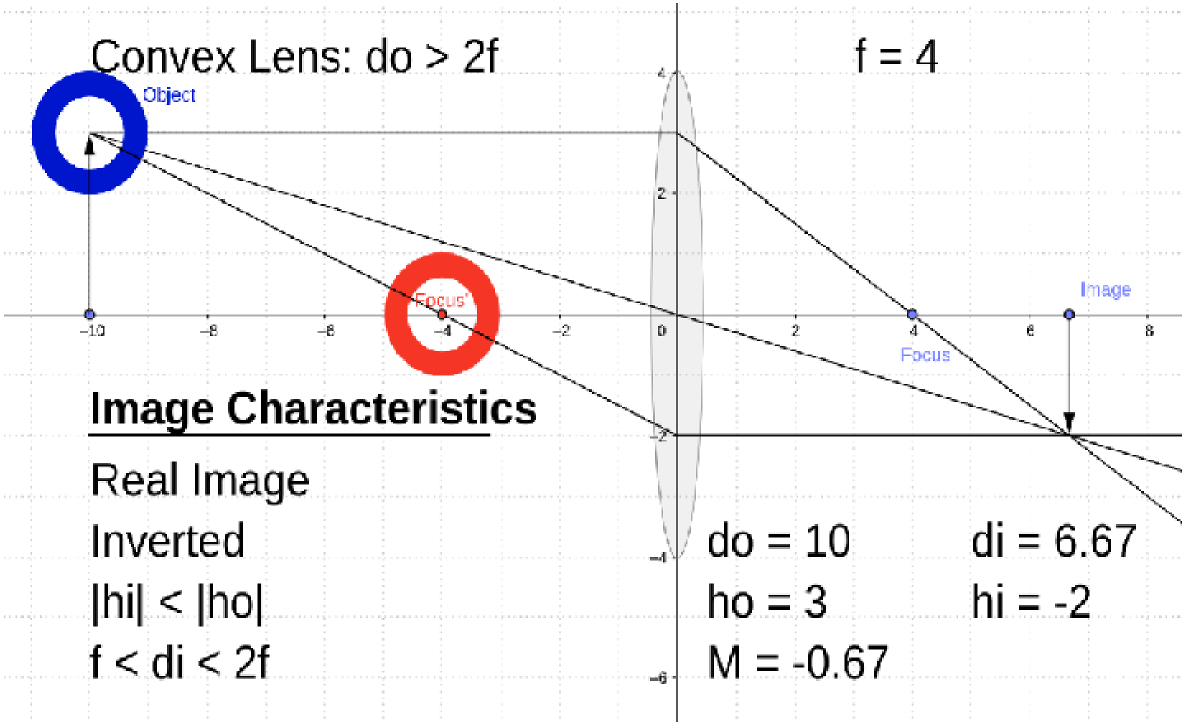


- Pinhole cannot be too small or too big
  - Too big: getting blur from overlapping of multiple light source
  - Too small: getting blur from diffraction
- Ideal pinhole size with diameter $\sim 2\sqrt{f\lambda}$
- Size is usually small for visible light and a reasonable size f $\Rightarrow$need long exposure time
- Use lenses!

Lens camera

Gaussian lens (thin lens) law

$$\frac{1}{i} + \frac{1}{o} = \frac{1}{f}$$



Convex Lens: do > 2f

f = 4

Image Characteristics

Real Image

Inverted

|hi| < |ho|

f < di < 2f

do = 10

ho = 3

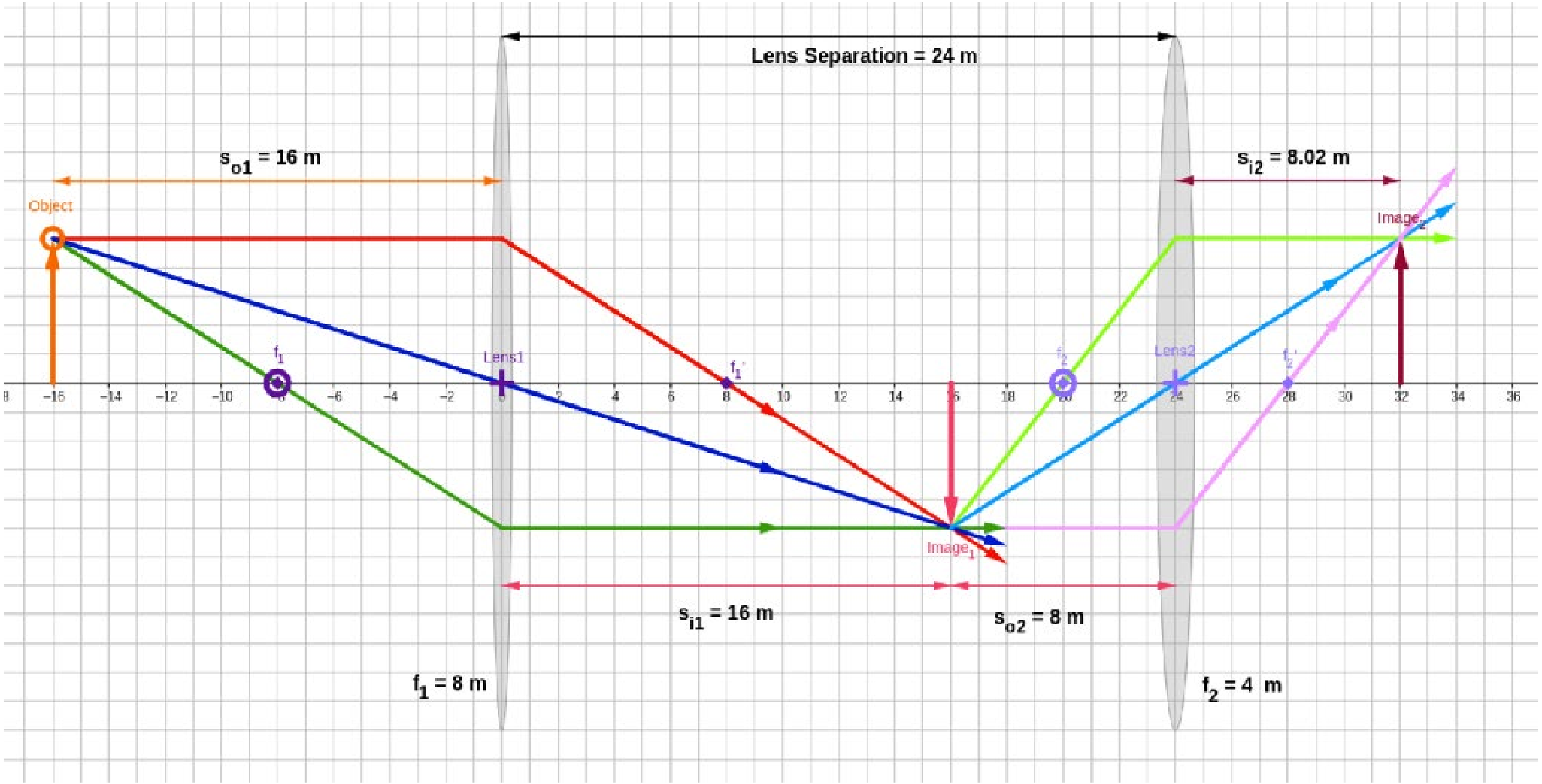
M = -0.67

di = 6.67

hi = -2

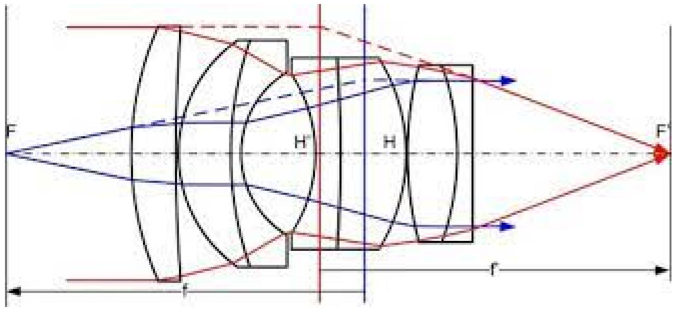
What is the magnification?

Two lens system



Compound lens system

- Can have 7-15 lenses in the system
- Can adjust “effective focal length” by varying lenses’ separations

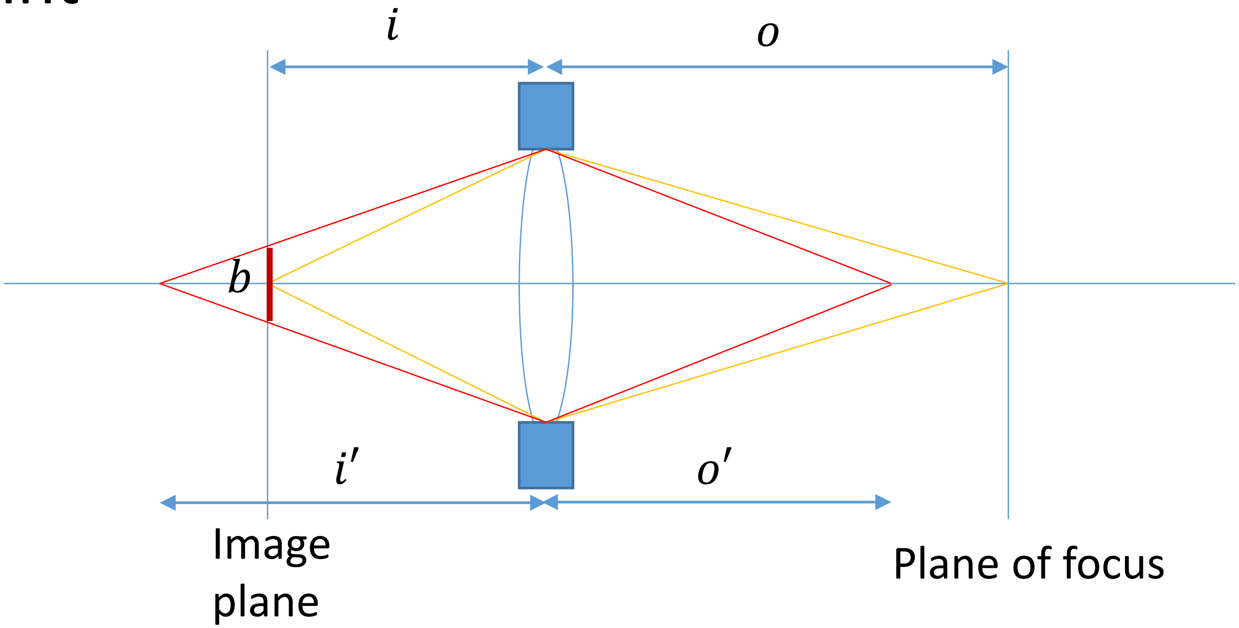


Aperature diameter and f-number (f-stop, f-ratio)

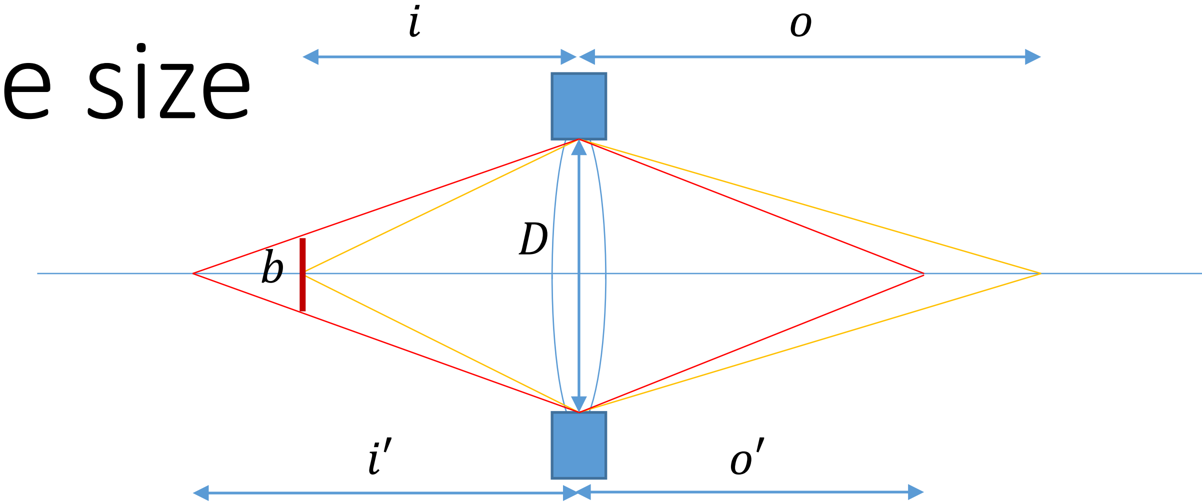
- Effective focal length f
- Aperture diameter D
- f-number: $N = \frac{f}{D}$
- E.g. 50 mm focal length, N=1.8, D =27.8 mm (fully open)

Blurred circle

- Unlike pinhole cameras, cameras with lenses cannot focus everywhere on the scene
- When a point lies outside the plane of focus in the scene, it maps to a blurred circle rather than a point



Blurred circle size



- $\frac{b}{D} = \left|\frac{i'-i}{i'}\right|$
- $b \propto D \propto \frac{1}{N}$

$$\frac{1}{i} + \frac{1}{o} = \frac{1}{i'} + \frac{1}{o'} = \frac{1}{\mathrm{f}} \Rightarrow i = \frac{of}{o-f}, i' = \frac{o'f}{o'-f}$$

$$\Rightarrow b = D\left|\frac{(o-o')\,f}{o'(o-f)}\right| = \frac{1}{N}\left|\frac{(o-o')f^2}{o'(o-f)}\right|$$

Depth of field

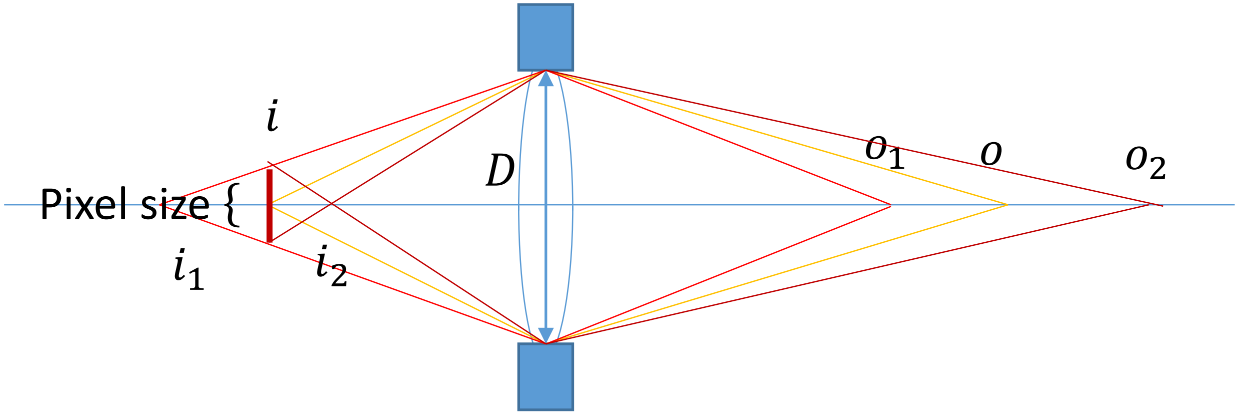
- Depth of field is the range of object distances over which the image is “sufficiently well” focused
  - i.e., the blurred circle is smaller than a pixel
- Note that the depth of field for pinhole camera is infinite

Computing depth of field

- Let pixel size be *c*. For convex lens, f, o_1, o_2, o are positive, so

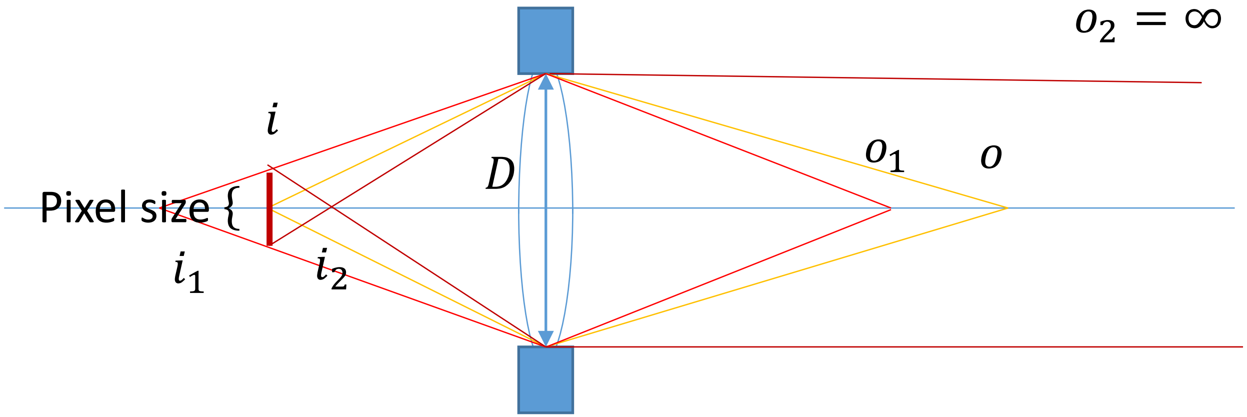
$$c = \frac{|(o - o_1)|f^2}{o_1(o - f)N} = \frac{|(o - o_2)|f^2}{o_2(o - f)N} = \frac{(o-o_1)f^2}{o_1(o-f)N} = \frac{(o_2-o)f^2}{o_2(o-f)N}$$

- Depth of field = $o_2 - o_1 = \dfrac{2of^2cN(o-f)}{f^4-c^2N^2(o-f)^2}$



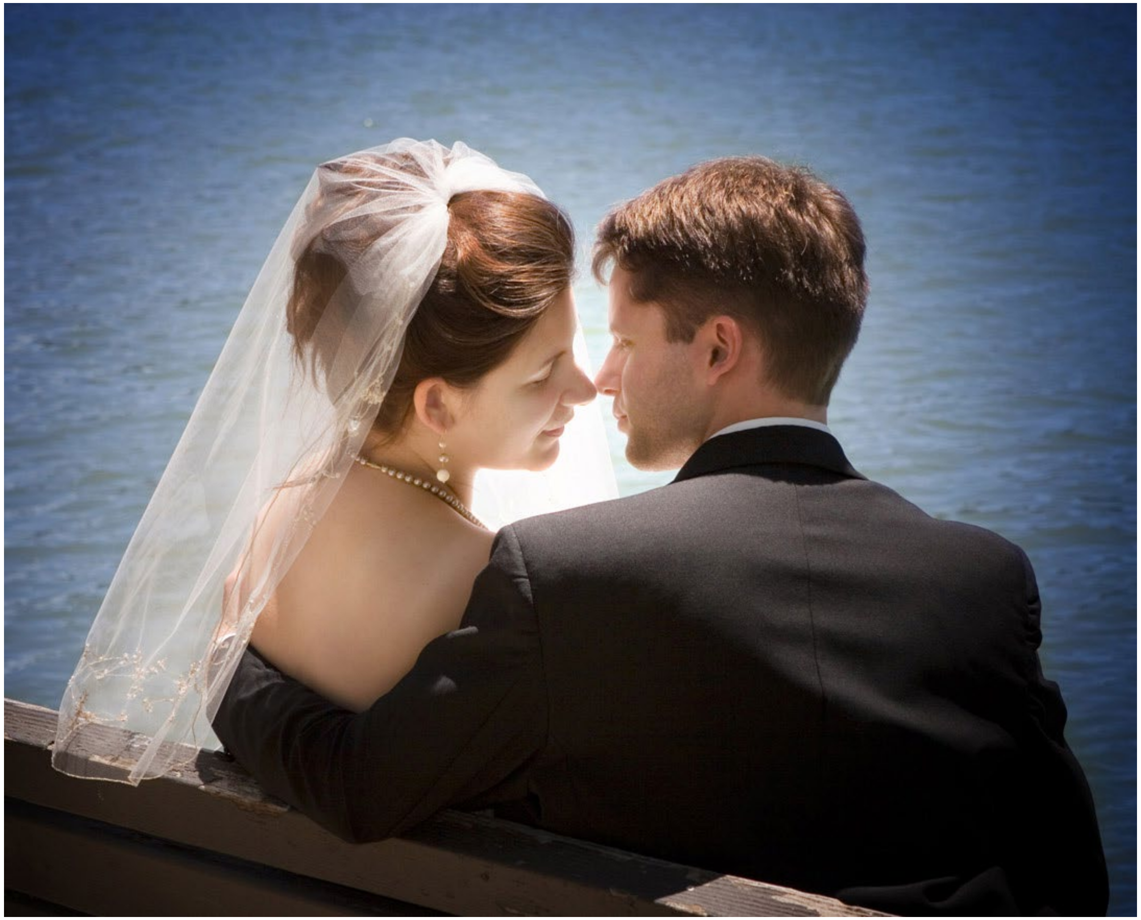
Hyperfocal distance

- It is convenient to set o_2 to infinity so that everything beyond certain range is in focus
- In this case, we call the respective o the hyperfocal distance
- Set $o_2 = \infty$, we have $c = \frac{f^2}{(o-f)N} \Rightarrow o = \frac{f(f+cN)}{cN}$

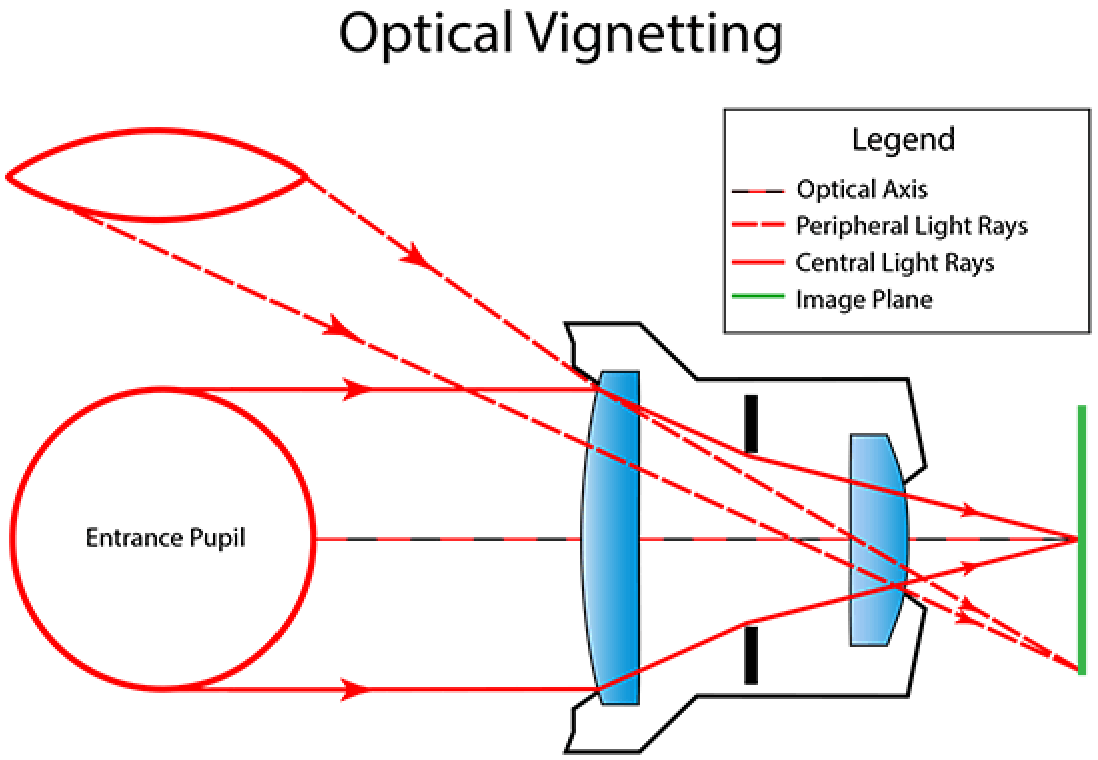


Camera distortion

Vignetting



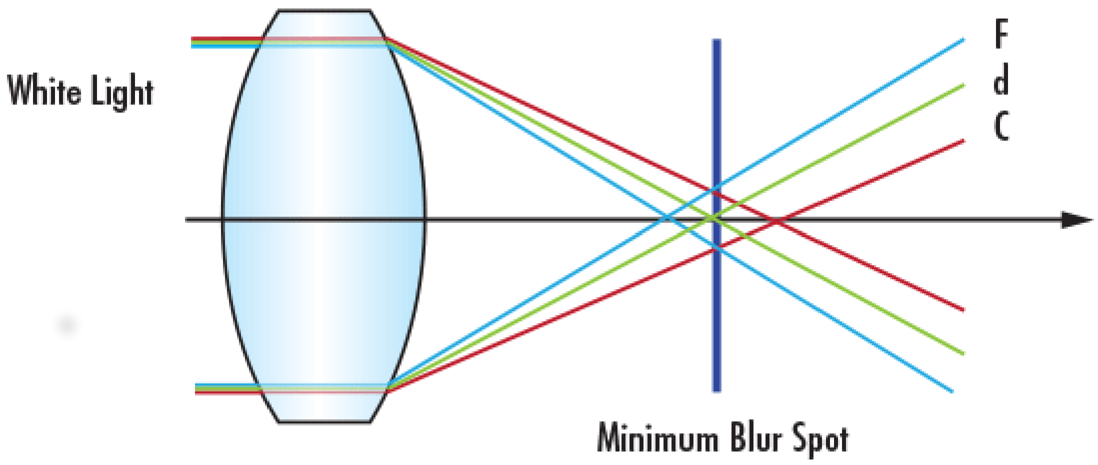
Vignetting


Optical Vignetting
Legend
Optical Axis
Peripheral Light Rays
Central Light Rays
Image Plane
Entrance Pupil

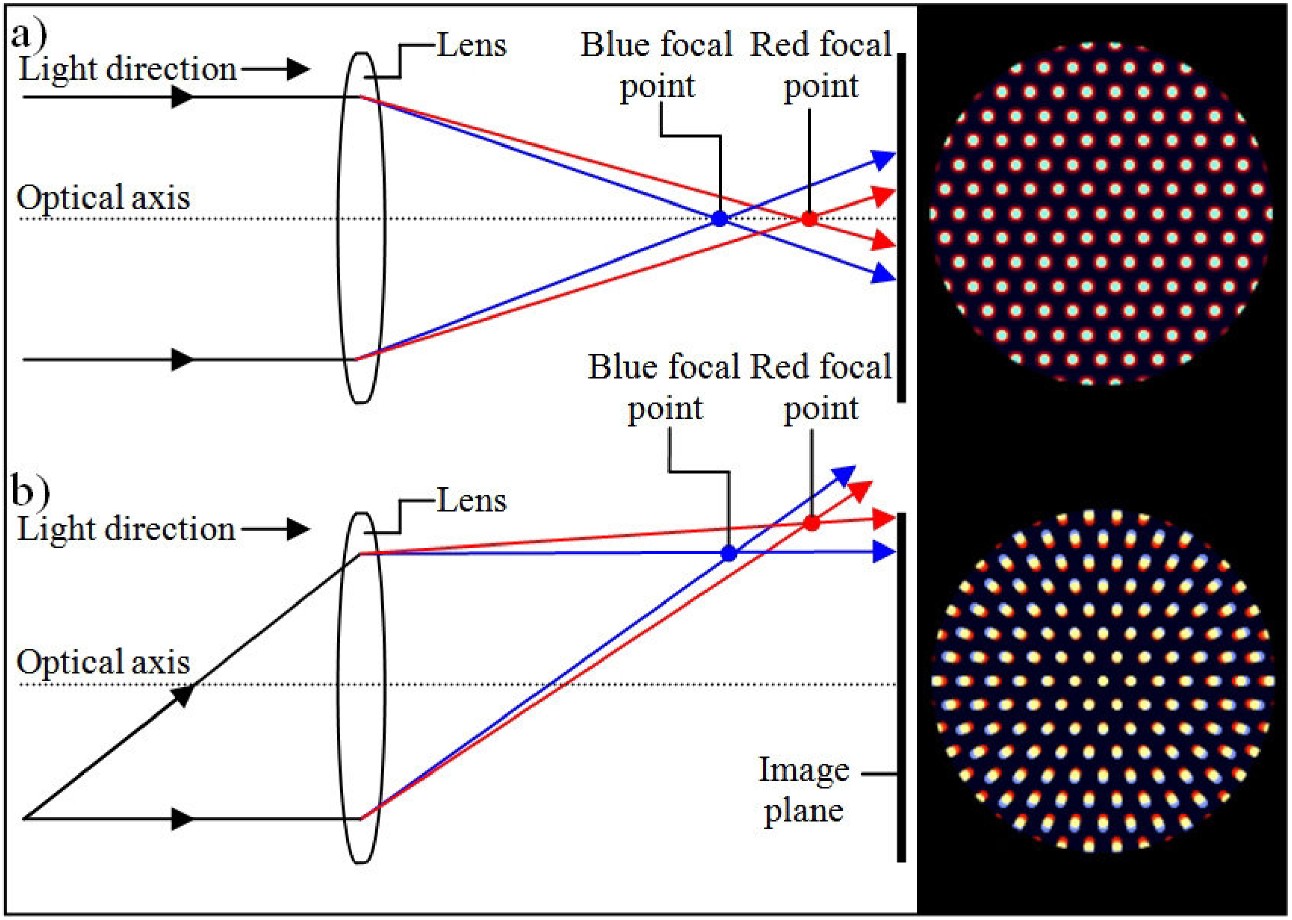
Chromatic aberration

CHROMA
A B C D E F G
H I J K L M
N O P Q R S
T U V W X Y Z
VectorStock®

Chromatic aberration

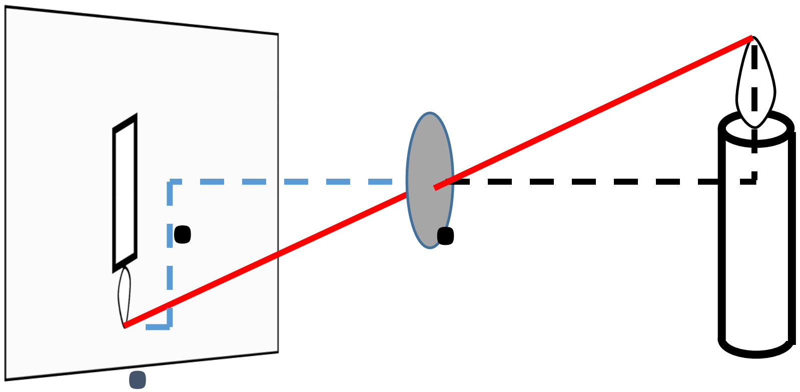


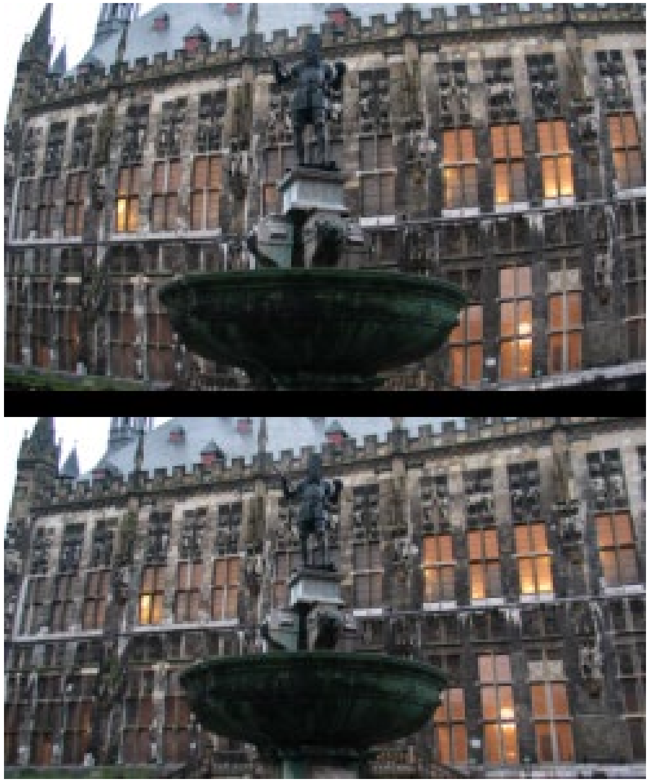
Chromatic aberration



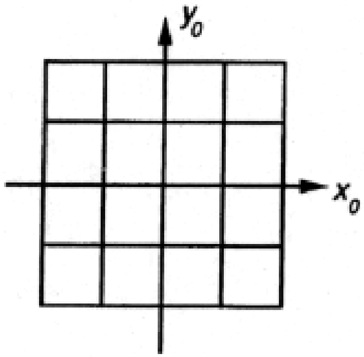
Radial distortion

Radial distortion is due to the imperfection of lens

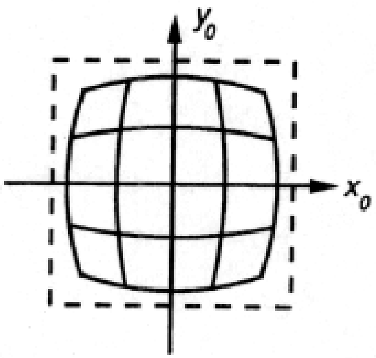




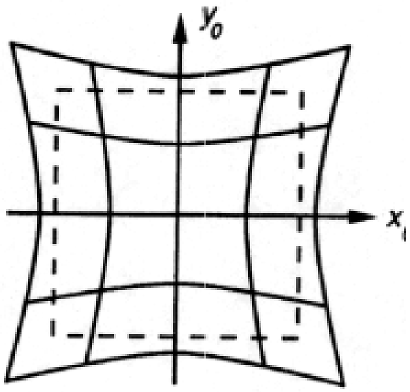
Corrected Barrel Distortion



No Distortion



Barrel Distortion



Pincushion Distortion

Image from Martin Habbecke

Radial distortion

- Radial distortion can be reduced by the following correction

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- *r* is the radial distance from the center of the scene
- The parameters can be estimated by shooting straight lines since a straight line is supposed to be preserved under perspective projection

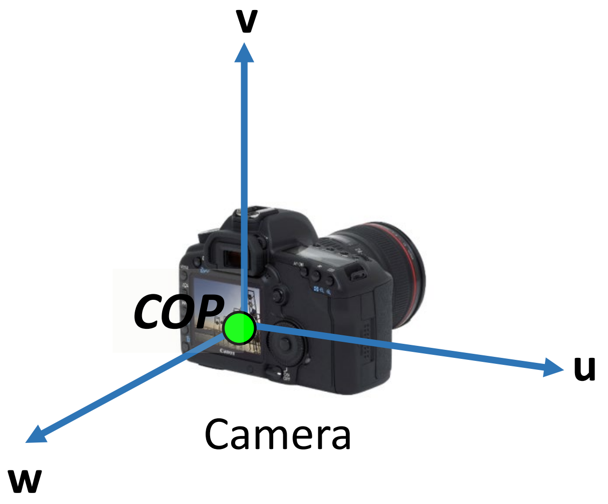
Camera matrix of pinhole camera

Camera parameters

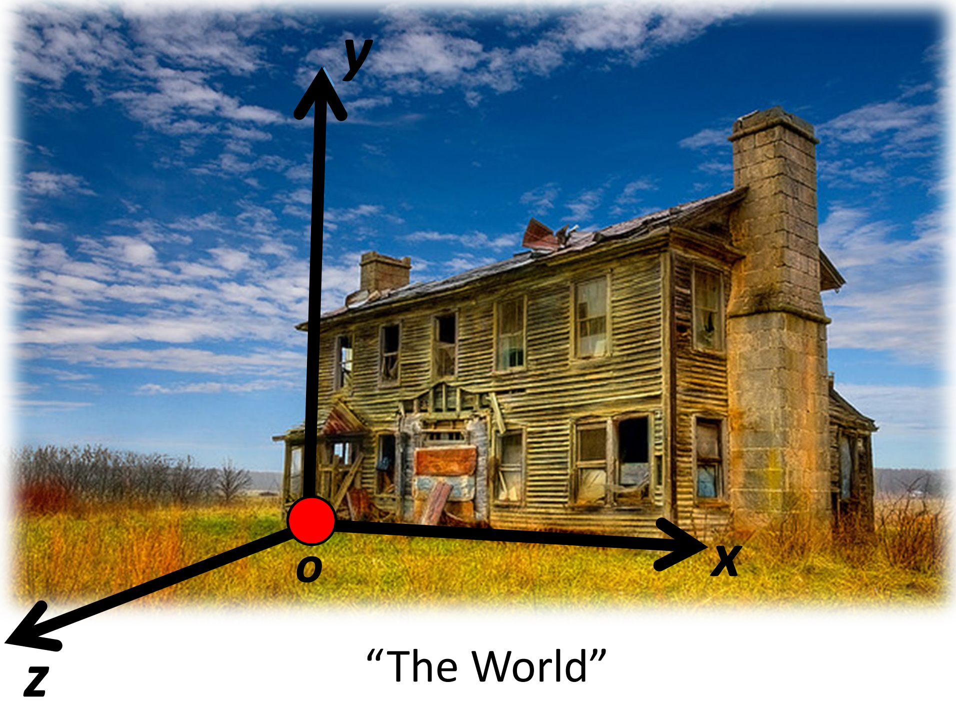
How many numbers do we need to describe a camera?

- We need to describe its *pose* in the world
- We need to describe its internal parameters

A Tale of Two Coordinate Systems



Camera



“The World”

Two important coordinate systems:

1. *World* coordinate system
2. *Camera* coordinate system

Camera parameters

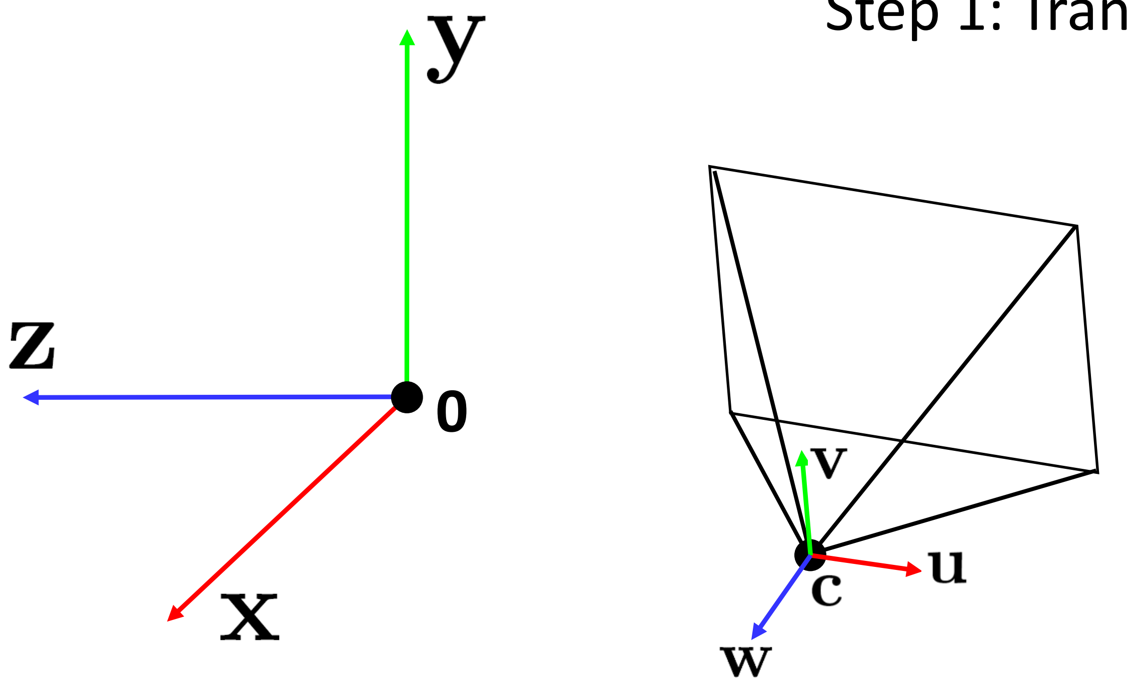
To project a point (*x*,*y*,*z*) in *world* coordinates into a camera

- First transform (*x*,*y*,*z*) into *camera* coordinates
- Need to know *extrinsics*
  - Camera position (in world coordinates)
  - Camera orientation (in world coordinates)
- Then project into the image plane
  - Need to know camera *intrinsics*
  - Coming soon
- These can all be described with matrices

Extrinsics

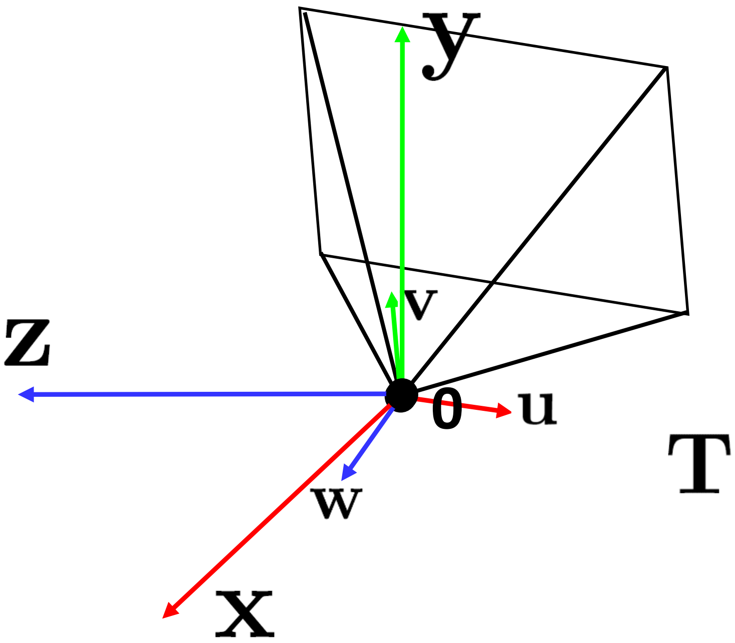
- How do we get the camera to “canonical form”?
    - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)

Step 1: Translate by -**c**



Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



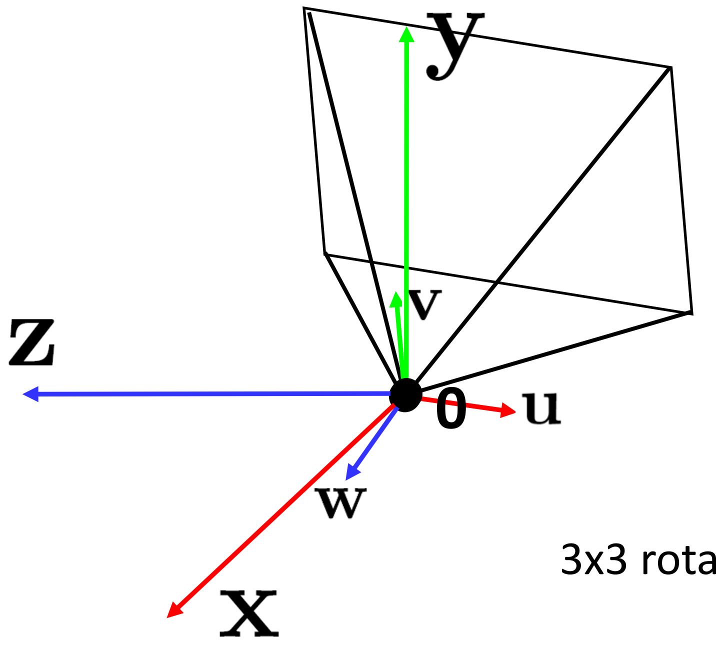
Step 1: Translate by -**c**

How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

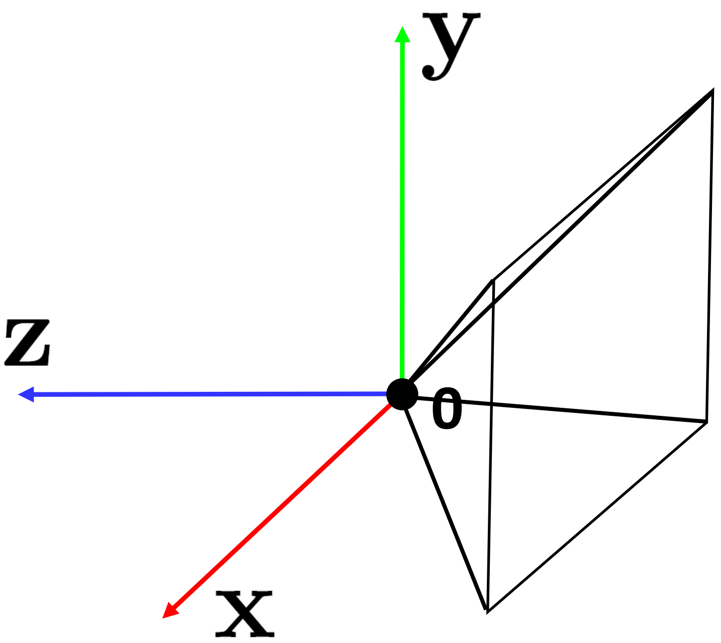
Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T & \\ \mathbf{v}^T & \\ \mathbf{w}^T & \\ & 1 \end{bmatrix}$$

3x3 rotation matrix

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T & \\ \mathbf{v}^T & \\ \mathbf{w}^T & \\ & 1 \end{bmatrix}$$

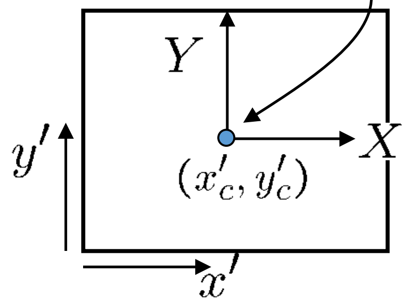
Camera parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called “extrinsics,” red are “intrinsics”

Projection equation

$$w\begin{bmatrix}U\\V\\1\end{bmatrix}=\begin{bmatrix}*&*&*&*\\*&*&*&*\\*&*&*&*\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}=\mathbf{\Pi X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

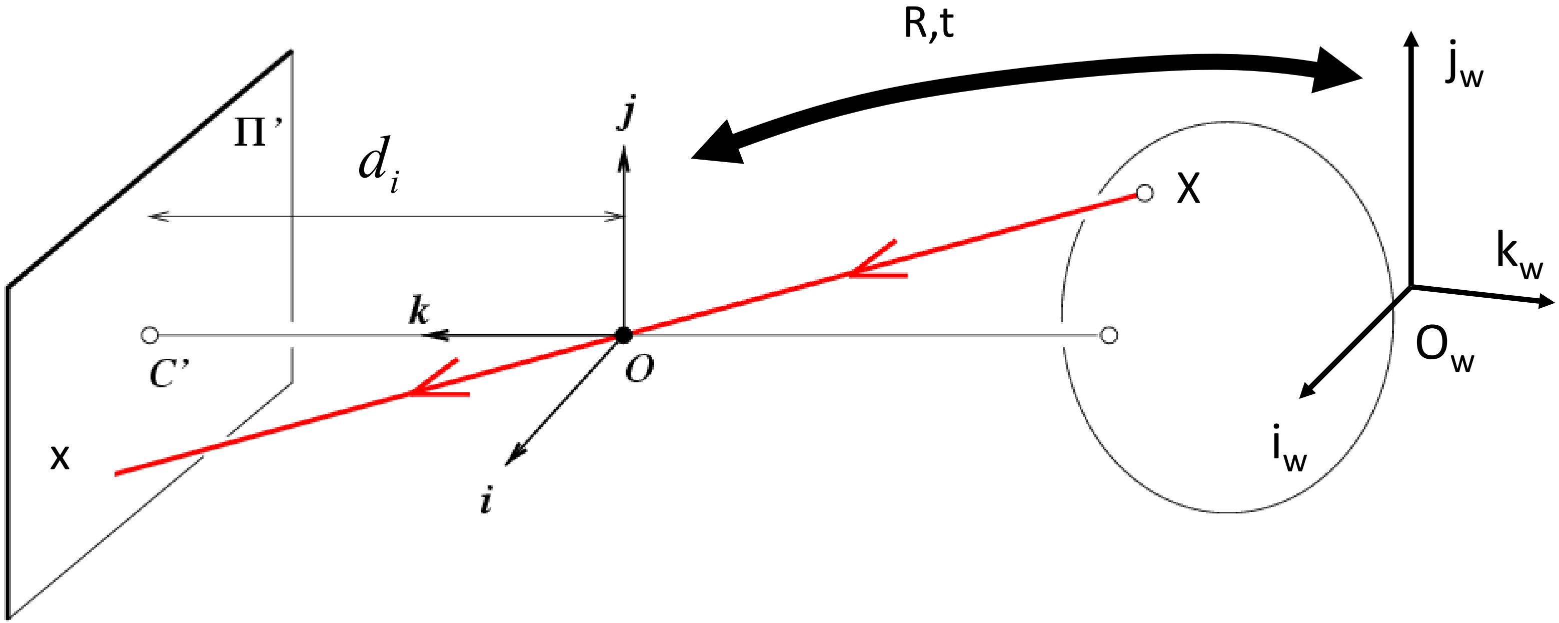
$$\mathbf{\Pi}=\overbrace{\underset{\text{intrinsics}}{\begin{bmatrix}*&*&*\\*&*&*\\*&*&*\end{bmatrix}}}^{\mathbf{K}}\overbrace{\underset{\text{projection}}{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}}\underset{\text{rotation}}{\begin{bmatrix}\mathbf{R}_{3x3}&\mathbf{0}_{3x1}\\\mathbf{0}_{1x3}&1\end{bmatrix}}\underset{\text{translation}}{\begin{bmatrix}\mathbf{I}_{3x3}&\mathbf{T}_{3x1}\\\mathbf{0}_{1x3}&1\end{bmatrix}}}^{[\mathbf{R}\quad\mathbf{RT}]}$$

identity matrix

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

Slide Credit: Savarese

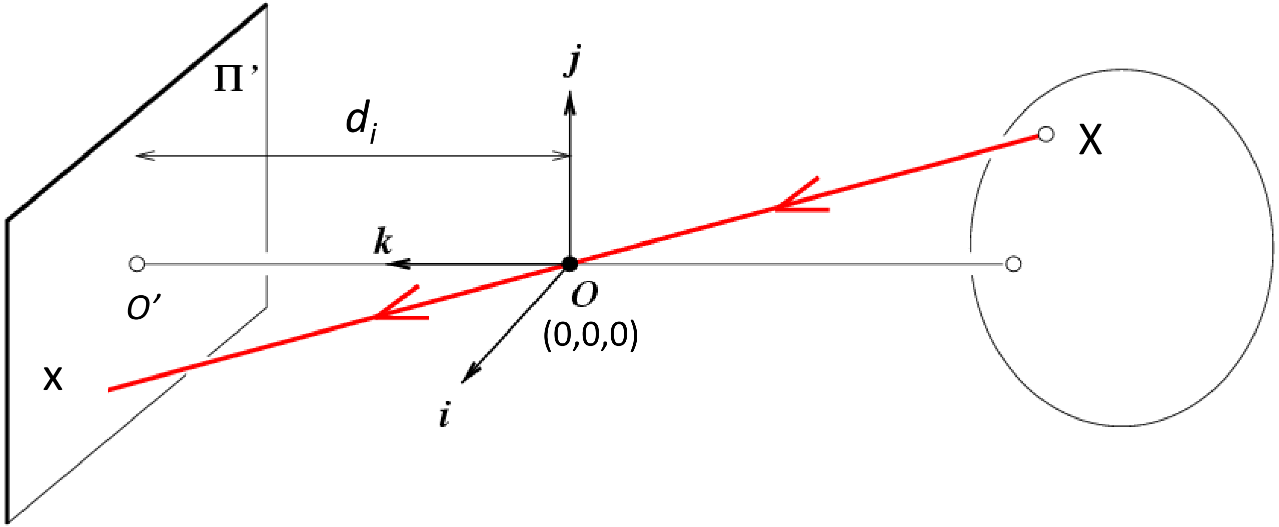
Camera (projection) matrix



$$\mathbf{x} = \mathbf{K}\left[\underbrace{\mathbf{R} \quad \overbrace{\mathbf{RT}}^{\mathbf{t}}}_{\text{Extrinsic Matrix}}\right]\mathbf{X}$$

x: Image Coordinates: (U,V,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Projection matrix (ignore extrinsics)



Intrinsic Assumptions

- Unit aspect ratio
- Optical center at (0,0)
- No skew

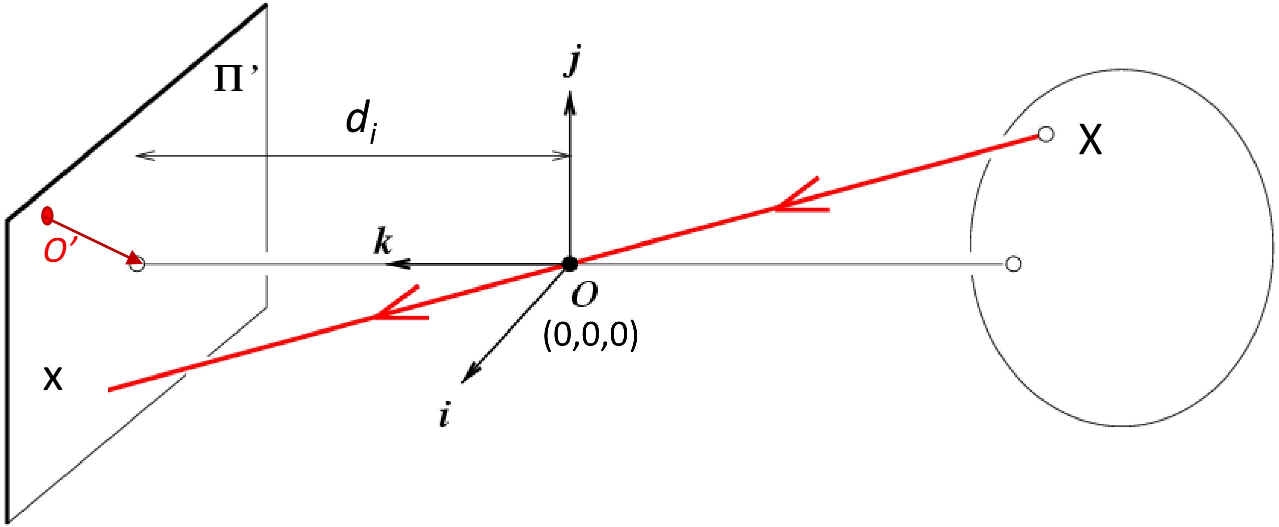
Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\left[\mathbf{I} \quad \mathbf{0}\right]\mathbf{X} \Rightarrow w\begin{bmatrix}U\\V\\1\end{bmatrix}=\left[\begin{matrix}\overbrace{\begin{matrix}-d_i&0&0\\0&-d_i&0\\0&0&1\end{matrix}}^{\mathbf{K}}&\begin{matrix}0\\0\\0\end{matrix}\end{matrix}\right]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Slide Credit: Savarese

Remove assumption: aligned optical center



Intrinsic Assumptions

- Unit aspect ratio
- Optical center at $-(U_0,V_0)$
- No skew

Extrinsic Assumptions

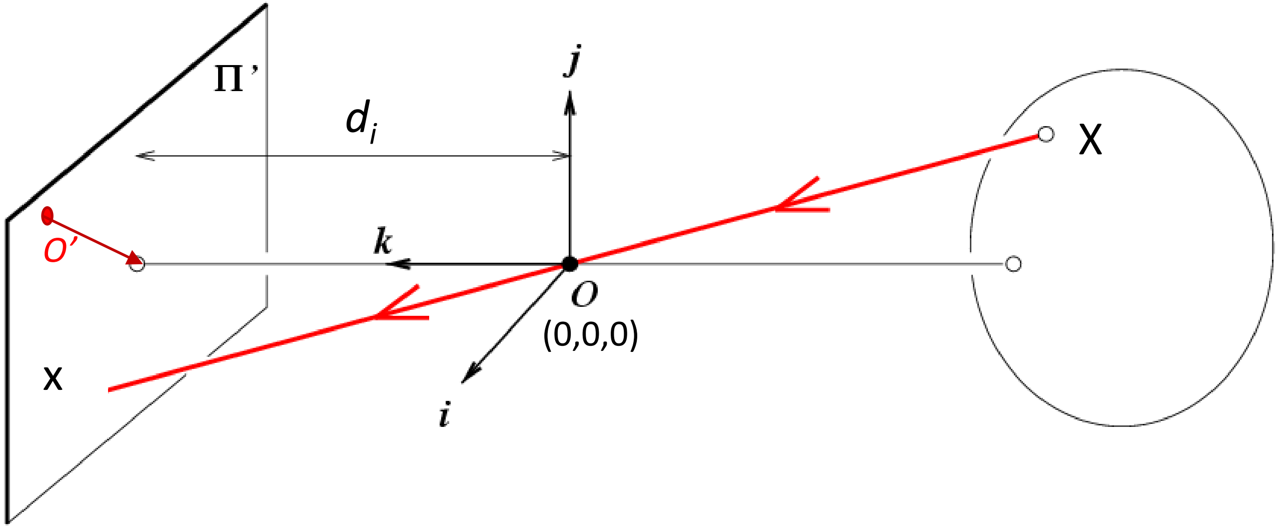
- No rotation
- Camera at (0,0,0)

K

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}U\\V\\1\end{bmatrix} = \left[\begin{array}{ccc|c}-d_i & 0 & U_0 & 0\\ 0 & -d_i & V_0 & 0\\ 0 & 0 & 1 & 0\end{array}\right]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Slide Credit: Savarese

Remove assumption: unit aspect ratio



Intrinsic Assumptions

- Optical center at $-(U_0,V_0)$
- No skew

Extrinsic Assumptions

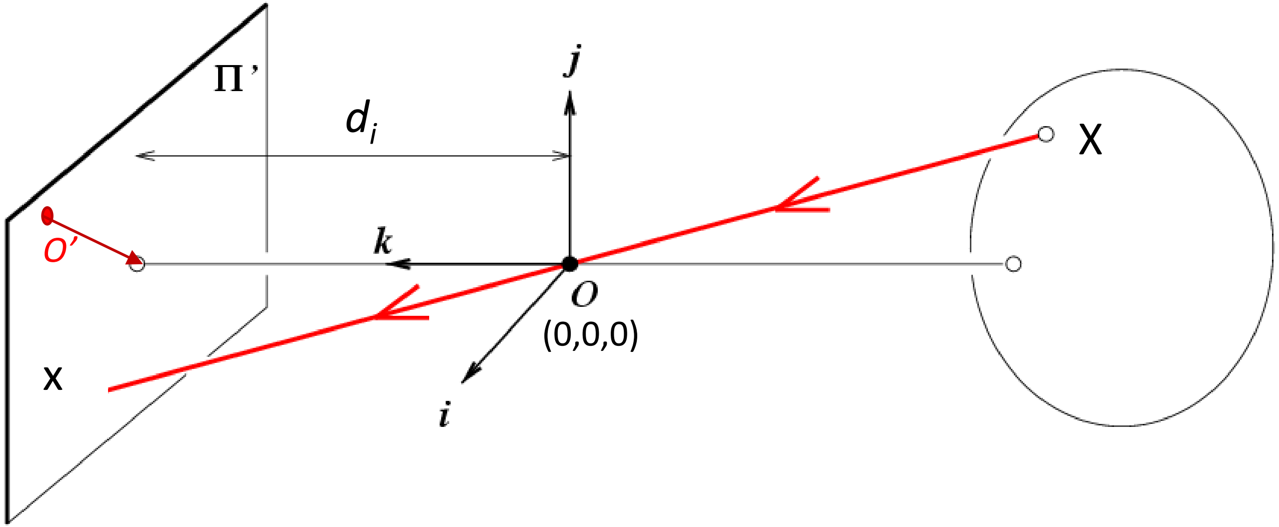
- No rotation
- Camera at (0,0,0)

K

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}U\\V\\1\end{bmatrix} = \begin{bmatrix}-d_i & 0 & U_0 & 0\\0 & -\alpha d_i & V_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Slide Credit: Savarese

Remove assumption: non-skewed pixels



Intrinsic Assumptions

- Optical center at $-(U_0,V_0)$

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}U\\V\\1\end{bmatrix} = \left[\begin{array}{c|c} \overbrace{\begin{matrix}-d_i & s & U_0\\ 0 & -\alpha d_i & V_0\\ 0 & 0 & 1\end{matrix}}^{\mathbf{K}} & \begin{matrix}0\\0\\0\end{matrix}\end{array}\right]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Slide Credit: Savarese

Summary

$$\mathbf{x} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \overbrace{\mathbf{RT}}^{\mathbf{t}} \end{bmatrix}\mathbf{X}$$

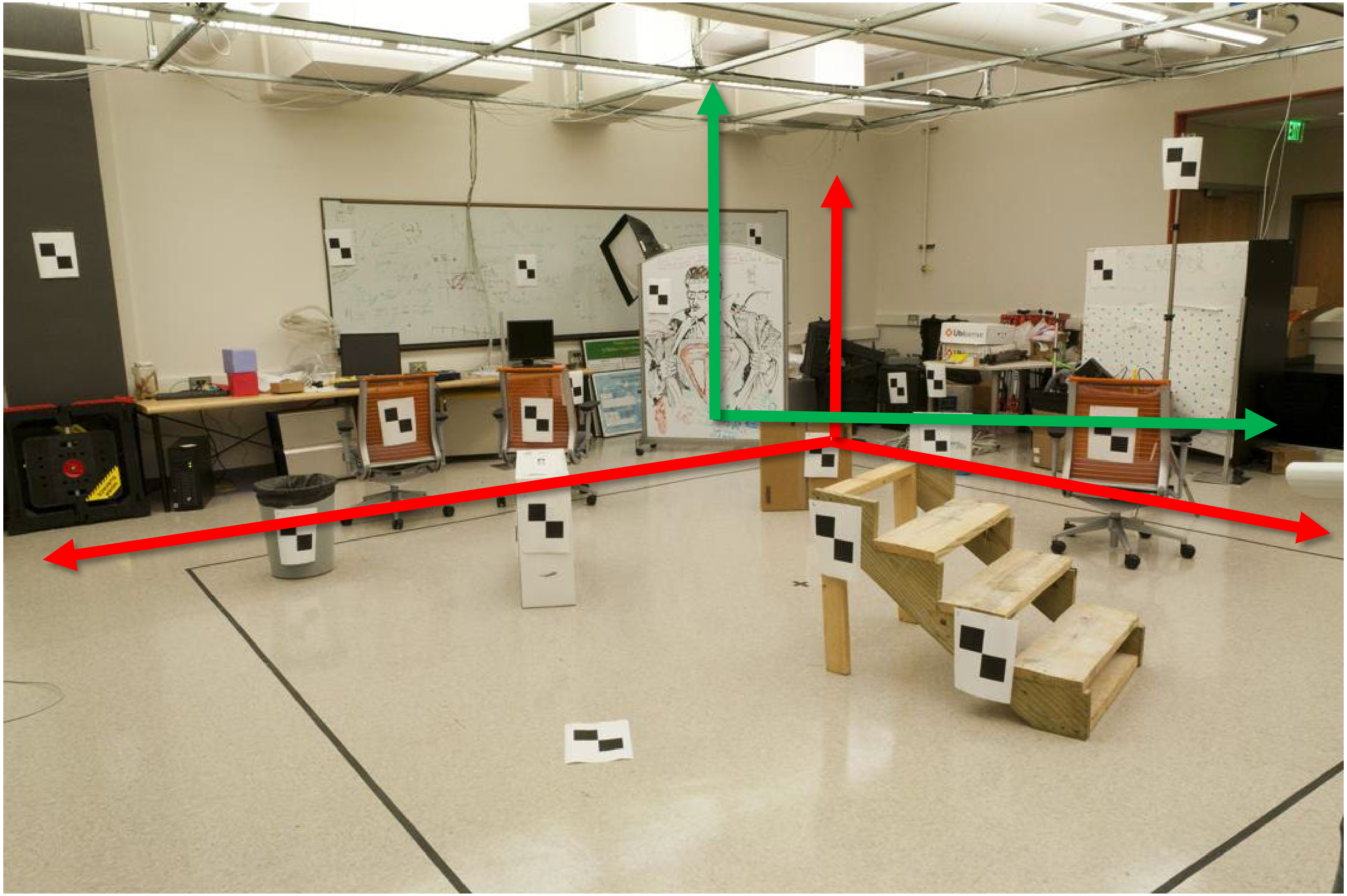
$$\Downarrow$$

$$w\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} -d_i & s & U_0 \\ 0 & -\alpha d_i & V_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

James Hays

Camera Matrix DEMO

World vs Camera coordinates

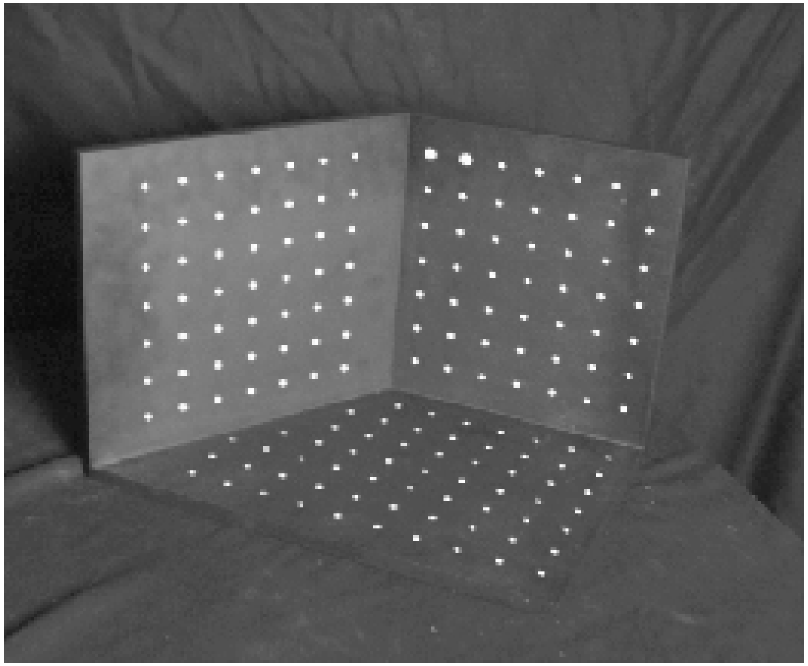


James Hays

Calibrating the Camera

Use an scene with **known** geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



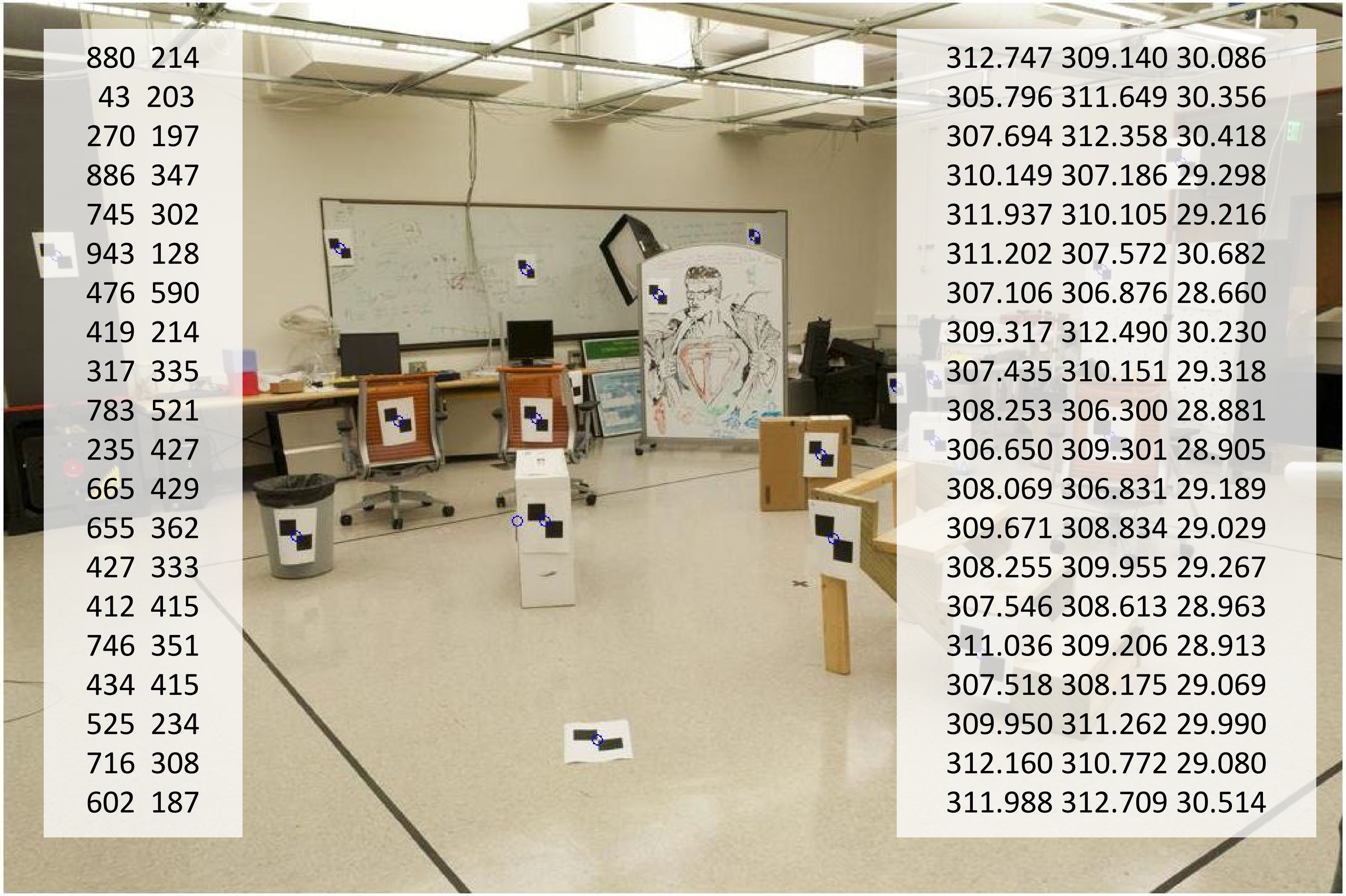
Known 2d image coords

Known 3d world locations

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \overset{\mathbf{M}}{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Unknown Camera Parameters

How do we calibrate a camera?



Known 2d image coords

880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

Known 3d world locations

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

James Hays

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

First, work out where X,Y,Z projects to under candidate **M.**

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

Two equations per 3D point correspondence

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

James Hays

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

Next, rearrange into form where all **M** coefficients are individually stated in terms of X,Y,Z,u,v.
-> Allows us to form lsq matrix.

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$
$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$
$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

Next, rearrange into form where all **M** coefficients are individually stated in terms of X,Y,Z,u,v.
-> Allows us to form lsq matrix.

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$

$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

- Solve for m's entries using total linear least-squares.

Ax=0 form

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix}}_{x} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

James Hays

A**x**=0

- Note that **x**=0 is a trivial solution and has to be avoided
- Consider instead

$$\begin{array}{l}\min_{\mathbf{x}} \ \|\mathbf{Ax}\| \\ \text{subject to } \|\mathbf{x}\| = 1\end{array} \quad \equiv \quad \begin{array}{l}\min_{\mathbf{x}} \ \mathbf{x}^T\mathbf{A}^T\mathbf{A}\,\mathbf{x} \\ \text{subject to } \|\mathbf{x}\| = 1\end{array}$$

Let $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$ be normalized eigenvectors of $\mathbf{A}^T\mathbf{A}$ with increasing eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_N$

Write $\mathbf{x} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_N\mathbf{u}_N$ with $\sum_{i=1}^{N} c_i^2 = 1$ and $c_i \geq 0$

We have $\|\mathbf{x}\| = 1$ is satisfied

$$\mathbf{x}^T\mathbf{A}^T\mathbf{Ax} = \left(\sum_{i=1}^{N} c_i\mathbf{u}_i^T\right)\mathbf{A}^T\mathbf{A}\left(\sum_{j=1}^{N} c_j\mathbf{u}_j\right) = \left(\sum_{i=1}^{N} c_i\mathbf{u}_i{}^T\right)\left(\sum_{j=1}^{N} c_j\lambda_j\mathbf{u}_j\right) = \sum_{i=1}^{N} c_i^2\lambda_i$$

$\|\mathbf{Ax}\|$ is minimized if we pick $c_1 = 1$ and $c_i = 0, \forall i > 1$ $\quad \therefore \mathbf{x} = \mathbf{u}_1$

SVD and eigen-decomposition

- Need to solve the eigen-decomposition problem of A^TA. But often it is better to solve SVD of A instead
- SVD: Singular value decomposition
- Every real matrix A can be written as USV^T, where U and V are orthogonal and S is diagonal
- Consider $A^TA=(VSU^T)USV^T=VS^2V^T$
  - That is, $(A^TA)V=VS^2$, V is eigenvector matrix of A^TA and S^2 is eigenvalue matrix of A^TA
- Instead of solving eigen-decomposition of A^TA, we can solve SVD of A instead

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

- Solve for m’s entries using total linear least-squares.

Ax=0 form

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix}}_{x} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

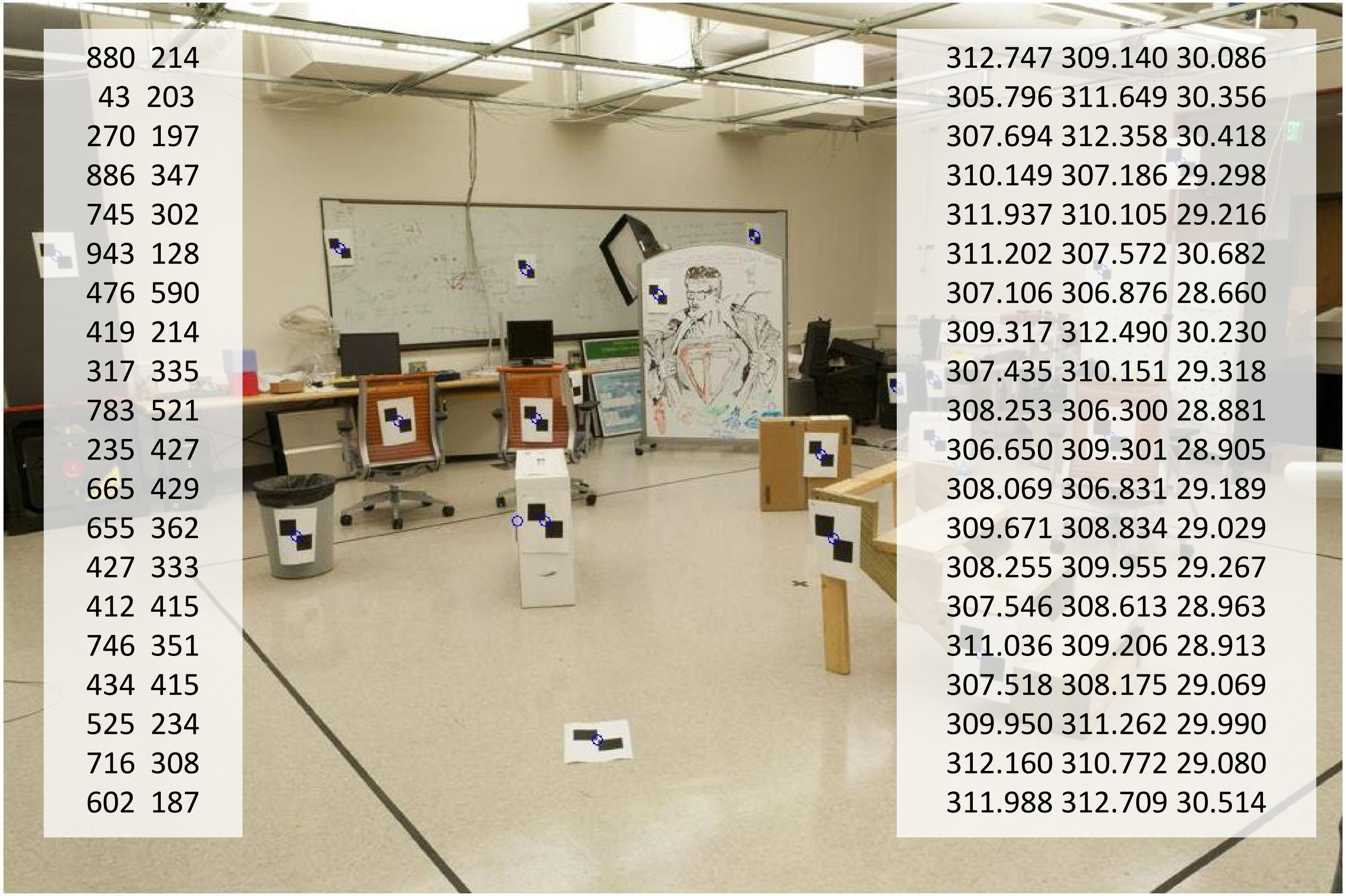
```
[U, S, V] = svd(A);
x = V(:,end);
M = reshape(x,4,3)';
```

James Hays

How do we calibrate a camera?

Known 2d image coords

Known 3d world locations



880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

James Hays

Known 2d image coords

Known 3d world locations

1st point (u_1, v_1) (X_1, Y_1, Z_1)

880	214	312.747	309.140	30.086
43	203	305.796	311.649	30.356
270	197	307.694	312.358	30.418
886	347	310.149	307.186	29.298
745	302	311.937	310.105	29.216
943	128	311.202	307.572	30.682
476	590	307.106	306.876	28.660
419	214	309.317	312.490	30.230
317	335	307.435	310.151	29.318
…		…..		

Projection error defined by two equations – one for *u* and one for *v*

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times312.747 & -880\times309.140 & -880\times30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times312.747 & -214\times309.140 & -214\times30.086 & -214 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Known 2d image coords

Known 3d world locations

2nd point

880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335

…

(u_2, v_2)

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318

…..

(X_2, Y_2, Z_2)

Projection error defined by two equations – one for *u* and one for *v*

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times 312.747 & -880\times 309.140 & -880\times 30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times 312.747 & -214\times 309.140 & -214\times 30.086 & -214 \\ 305.796 & 311.649 & 30.356 & 1 & 0 & 0 & 0 & 0 & -43\times 305.796 & -43\times 311.649 & -43\times 30.356 & -43 \\ 0 & 0 & 0 & 0 & 305.796 & 311.649 & 30.356 & 1 & -203\times 305.796 & -203\times 311.649 & -43\times 30.356 & -203 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1_n & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

How many points do I need to fit the model?

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Degrees of freedom?

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} \left.\underbrace{\begin{matrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{matrix}}_{\text{Think 3}}\right\} & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Think 3:

- Rotation around x
- Rotation around y
- Rotation around z

How many points do I need to fit the model?

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\Downarrow$$

Degrees of freedom?

$$w\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = \overset{5}{\begin{bmatrix}\alpha & s & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix}}\overset{6}{\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\ r_{21} & r_{22} & r_{23} & t_y\\ r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

M is 3x4, so 12 unknowns, but projective scale ambiguity – 11 deg. freedom.
One equation per unknown -> 5 1/2 point correspondences determines a solution (e.g., either u or v).

More than 5 1/2 point correspondences -> overdetermined, many solutions to **M**.
Least squares is finding the solution that best satisfies the overdetermined system.

Why use more than 6? Robustness to error in feature points.

Summary

$$\overbrace{\begin{bmatrix}* & * & * \\ * & * & * \\ * & * & *\end{bmatrix}}^{\mathbf{K}}\overbrace{\begin{bmatrix}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1\end{bmatrix}\begin{bmatrix}\mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1\end{bmatrix}}^{[\mathbf{R} \quad \mathbf{RT}]}$$

$$\mathbf{x} = \mathbf{K}\overbrace{\begin{bmatrix}\mathbf{R} & \underbrace{\mathbf{RT}}_{t}\end{bmatrix}}\mathbf{X}$$

$$\Downarrow$$

$$w\begin{bmatrix}U \\ V \\ 1\end{bmatrix} = \begin{bmatrix}-d_i & s & U_0 \\ 0 & -\alpha d_i & V_0 \\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}X \\ Y \\ Z \\ 1\end{bmatrix}$$

Can we factorize M back to K [R | t]?

- Yes!
- We can directly solve for the individual entries of K [R | t].

James Hays

Can we factorize M back to K [R | t]?

- Yes!
- We can also use *RQ* factorization (not QR)
  - R in RQ is not rotation matrix R; crossed names!
- *R* (right diagonal) is K
- *Q* (orthogonal basis) is R.
- t, the last column of [R | t], is inv(K) * last column of M.
  - See http://ksimek.github.io/2012/08/14/decompose/ for more details

James Hays

Recovering the camera center

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$w\begin{bmatrix}u\\ v\\ 1\end{bmatrix}=\begin{bmatrix}\alpha & s & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix}\left[\begin{matrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{matrix}\ \overset{\mathbf{t}}{\boxed{\begin{matrix}t_x\\ t_y\\ t_z\end{matrix}}}\right]\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

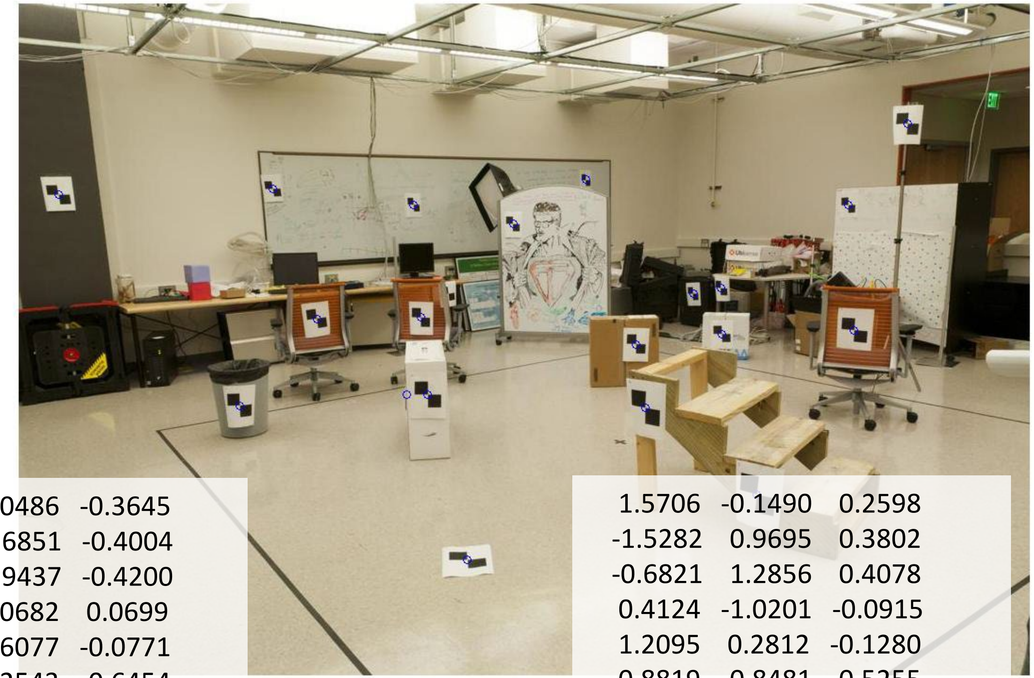
$$\begin{bmatrix}su\\ sv\\ s\end{bmatrix}=\left[\underbrace{\boxed{\begin{matrix}* & * & *\\ * & * & *\\ * & * & *\end{matrix}}}_{Q}\ \overset{m_4}{\boxed{\begin{matrix}*\\ *\\ *\end{matrix}}}\right]\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}$$

$t= K^{-1}\ m_4$

$T=R^{-1}t=R^{-1}\ K^{-1}\ m_4$

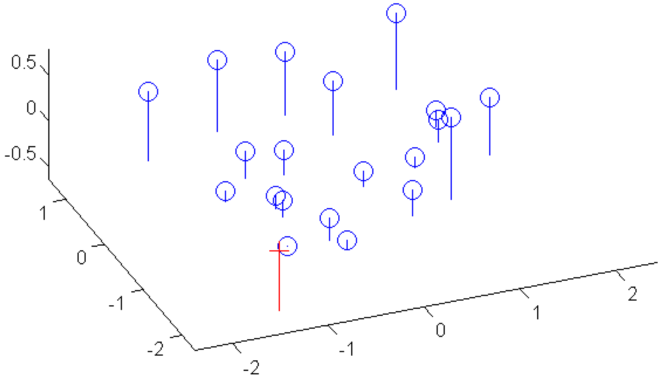
James Hays

Estimate of camera center



1.0486	-0.3645
-1.6851	-0.4004
-0.9437	-0.4200
1.0682	0.0699
0.6077	-0.0771
1.2543	-0.6454
-0.2709	0.8635
-0.4571	-0.3645
-0.7902	0.0307
0.7318	0.6382
-1.0580	0.3312
0.3464	0.3377
0.3137	0.1189
-0.4310	0.0242
-0.4799	0.2920
0.6109	0.0830
-0.4081	0.2920
-0.1109	-0.2992
0.5129	-0.0575
0.1406	-0.4527

1.5706	-0.1490	0.2598
-1.5282	0.9695	0.3802
-0.6821	1.2856	0.4078
0.4124	-1.0201	-0.0915
1.2095	0.2812	-0.1280
0.8819	-0.8481	0.5255
-0.9442	-1.1583	-0.3759
0.0415	1.3445	0.3240
-0.7975	0.3017	-0.0826
-0.4329	-1.4151	-0.2774
-1.1475	-0.0772	-0.2667
-0.5149	-1.1784	-0.1401
0.1993	-0.2854	-0.2114
-0.4320	0.2143	-0.1053
-0.7481	-0.3840	-0.2408
0.8078	-0.1196	-0.2631
-0.7605	-0.5792	-0.1936
0.3237	0.7970	0.2170
1.3089	0.5786	-0.1887
1.2323	1.4421	0.4506



Calibration with non-linear methods

- Linear calibration
  - Advantages
    - Easy to formulate and solve
    - Provides initialization for non-linear methods
  - Disadvantages
    - Doesn’t directly give you camera parameters
    - Doesn’t model radial distortion
    - Can’t impose constraints, such as known focal length
- Non-linear calibrations
  - Define error as difference between projected points and measured points
  - Minimize error using Newton’s method or other non-linear optimization

James Hays

OpenCV Calibration Demo